cas-serializability-test

По условию задано множество операций $\{CAS_i(res_i, exp_i, new_i, start_i, end_i)\}_{i=1}^{K}$, где res - результат исполнения операции ($success/fail$), exp - ожидаемое значение, new - значение для записи, $start$ и end - соответственно времена начала и конца операции.

Так как нас интересует сериализация исполнения, рассмотрим определение:

> A history is **serializable** if it is equivalent to one in which transactions appear to execute sequentially, i.e., without interleaving. ("Linearizability: A Correctness Condition for Concurrent Objects").

Также в статье упоминается термин **strict serializability,** который учитывает временной порядок над операциями. Для наших целей временные отметки оказываются излишними, поэтому в дальнейшем мы будем использовать сокращенное обозначение$CAS_i(res_i, exp_i, new_i)$.

Пусть нам известно начальное значение регистра на момент инициализации ($init$), и конечное значение ($result$).

Пусть (N, A) - ориентированный граф. Положим множество вершин $N = \{new_i\}_{i=1}^{K} \cup \{expected_i\}_{i=1}^{K} \cup \{init, result\}$ - все возможные значения регистра, упоминаемые в данном множестве операций, вместе с начальным и конечным состояниями.

Между вершинами a и b есть ребро, если существует успешный $CAS(res, exp, new)$, где $res = success$, $exp = a, new = b$.

Неудачную операцию $CAS(res_f, exp_f, new_f)$, где $res_f = fail$, можно представить в виде петли у произвольной вершины из N, значение в которой не равно exp_f (чтобы операция в принципе была неуспешной), т.к. по своей сути неудачная операция не изменяет значение регистра.

Покажем, что такая вершина всегда есть. Предположим, что все вершины из N имеют значения, равные exp_f - это означает, что граф состоит из единственной вершины со значением exp_f, то есть $exp_f = init = result = exp_i = new_i$ для всех допустимых i. Очевидно, в регистре могут лежать только значения из N, вершин с другими значениями быть не может; выходит, что неуспешная операция $CAS(res_f, exp_f, new_f)$ в таких условиях не может существовать. Выбор подходящей для петли вершины не влияет на корректность алгоритма.

Произвольный путь в полученном графе состоит из ребер, которые представляют собой операции $CAS_i(res_i, exp_i, new_i)$. Если у нас получится найти путь из вершины $init$ до вершины $result$, содержащий все ребра в графе - мы получим требуемое последовательное исполнение транзакций. Эта последовательность ребер (CAS-операций) будет являться эйлеровым путем в графе из вершины $init$ в вершину $result$.

Эту задачу можно решить методом поиска пути Эйлера из вершины $init$ в $result$ за полиномиальное время.

Ориентированный граф содержит Эйлеров путь тогда и только тогда, когда все ребра принадлежат одной компоненте слабой связности и:

- для любой вершины $deg^+(v) = deg^-(v)$ (получим Эйлеров цикл)
- для одной вершины $deg^+(v_1) = deg^-(v_1) + 1$, для еще одной $deg^+(v_2) + 1 = deg^-(v_2)$, для остальных же выполняется условие $deg^+(v) = deg^-(v)$

Алгоритм решения:

1. Строим граф на основе заданных операций $CAS_i(res_i, exp_i, new_i)$
2. Проверяем, лежат ли все ребра в одной компоненте слабой связности - $O(N + A)$
3. Проверяем инварианты для полустепеней входа-выхода у вершин - $O(N)$

В пункте 3 возможны 2 ситуации:

- для любой вершины $deg^+(v) = deg^-(v)$. Тогда дополнительно нужно проверить, что $init == final$, а также что вершина $init$ принадлежит той единственной компоненте слабой связности (достаточно проверить, что $init$ не изолирована), если существуют ребра. Если ребер нет, достаточно проверки $init == final$.
- для одной вершины $deg^+(v_1) = deg^-(v_1) + 1$, для еще одной $deg^+(v_2) + 1 = deg^-(v_2)$, для остальных же выполняется условие $deg^+(v) = deg^-(v)$. В этом случае нужно проверить, что в вершине v_1

лежит значение $init$, а в вершине v_2 - значение $result$. Случай изолированности вершин рассматривать не нужно, т.к. он невозможен по условиям этой ситуации - хотя бы одна из полустепеней вершины больше нуля, а значит вершина не изолирована.

Если все условия эйлеровости выполнены, последним шагом можно построить Эйлеров путь, используя, например, алгоритм Хирхольцера - $O(N + A)$.

Имплементация: https://github.com/hardworkar/cas-serializability-test