Generating interesting tests using Gibbons[[1]](#footnote-1)

1. Randomly generate (all?) distinct value invocation sets with the following parameters
   1. N threads
   2. V values 🡺 I invocations = 2V (distinct values).
   3. V values
2. Trace should follow some guidelines
   1. Value pair invocation is **not** restricted to the same thread, but is not forbidden either.
   2. If a value pair invocation happens in the same thread, it should be in order. Otherwise it’s a trivial infringement on the ADT.
   3. , i.e. at least two invocations per thread. Otherwise, also trivial.
   4. Using distinct values on queues/stacks means that every *pop* has exactly one *push* so a weight system is not relevant.
3. Enumerate the threads and add “happens before” relation according to the values pairs thus generating a trace.
4. Use Gibbons’ algorithm to validate whether the trace is valid.   
   Keep **invalid** traces using *assume* rather than *assert* on (all?) *pop* values.
5. Run tests on a given implementation. If any run **passes** on any test, it has reached an undesirable state.

Notes:

1. Is distinct value a reasonable limitation?
2. Do we only keep invalid test cases or also valid ones?
3. There are probably more trivial behaviors we need to eliminate to make sure our tests are interesting.
4. Assuming we find bugs (known or new) we can play with the parameters and optimize them considering both performance and the chance to find the bugs.

GENMC:

1. Test with included file, includes should be

Action items:

1. Implement Gibbons, with trace inputs and YES/NO output
2. Implement tracing algorithm:
   1. Given a set of operations (with values)
   2. While Gibbons(trace) == YES
      1. Add pair to partial order in trace  
         Partial order received after adding a pair should be\*
         1. Acyclic
         2. Transitive
   3. If Gibbons(trace) == NO
      1. Check all subsets to find minimal

\*Keep partial order has its own limitations

Violat:

1. Weighted random execution *schema* generation. Weights are assigned to make it more probable to generate ADT “writes” (weight 3) over “reads” (weight 1).
2. Schema generation is parametrized with the defaults being:
   1. 2 threads
   2. 2 integer values
   3. 6 invocations in total
3. Generates a valid result set *per schema* but interleaving the invocations and statically analyzing the possible sequential results based on the “happens before” relation.
4. Runs the static test many times and matches results against the valid results set generated before the run. If a run ends in a result that doesn’t appear in the set, it is flagged as a bug.

Notes:

1. Considering that they only use 2 values, interleaving returned values over the schema will not affect performance that much.
2. There is no guarantee on how “interesting” their tests are.

1. Gibbons - Validates distinct value traces for queues or stacks. [↑](#footnote-ref-1)