Proving a concurrent datatype's correctness is usually more complicated than proving for a single-threaded data structure. In addition to the state explosion when more and more processes are involved, scheduling is one particular reason that proving is difficult. For example, two processes are enqueuing their data to a concurrent queue. The process calling earlier may be descheduled and interact with the queue after the other process.

The linearizability test from [TODO:reference] is one way to prove that a sequence of process interactions is correct using only trivially measurable information - pairs of calling and returning time of every function call made by every process. From the history of function calls, The linearizability test tries to find a synchronization point for each call, where the call appears to make an effect. In addition, the series of synchronization points should satisfy the specification of the concurrent datatype.

In order to show that a system using a concurrent datatype is correct, the linearizability test works by running the system many times and checking the traces. In this essay, we use CSP models and the linearizer framework to prove the correctness of a concurrent datatype.

CSP is ... Trace is ... Trace refinement is ... FDR is ...

In the CSP model, the history of all function call is represented using related events. Call.t.op.p represents a process calls function op with parameter p. Return.t.op.ans represents a process returns from calling op with return value ans.

In CSP, trace refinement is a valuable tool to analyze the history of function calls described in the linearizability test. If the testing system, which produces possible function calls and function returns in CSP, refines a CSP specification of the system, then the testing system cannot do anything incorrect and thus must be correct. One way to construct the specification is to use the linearizer framework.

The linearizer has two core components - the linearizer process and the specification process for the Sync event. The linearizer process generates endless pairs of Call and Return events with its synchronization point for a specific process. When all linearizer processes, together with the specification process for Sync, run in parallel, the combined process produces all possible histories in CSP. Figure 1 shows how to create the combined processes in FDR. In Section ... we shall see more concrete instances of the linearizer framework.

```
(( || me: All • [LinEvents(All,me)]  Lin(All,me)) [|{|Sync|}|] Spec) \ {|Sync|}
```

Figure 1: Creation of the specification from the linearizers