Chapter 3
Abstract Fault Tolerance:
A Model-Theoretic Approach to Fault Tolerance and Fault Compensation without Error Correction

Leo Marcus
The Aerospace Corporation, USA

ABSTRACT

This paper explores the concept of "pure fault tolerance"--when a system can fulfill its computational goal, even though some of its components do not fulfill theirs. This leads to a natural statement of the composition problem for a very general concept of architecture: to what extent are the properties of a system based on that architecture a function of the properties of the components? The author explores several variations on pure fault tolerance, none of which utilize fault detection or error correction.

1. INTRODUCTION

What to do about "errors" or "faults" is an important concern for the design and operation of computer systems, especially for potential quantum computer implementations.[1] Possible mitigation methods include avoidance, correction, compensation, and tolerance. These often involve fault detection.

This whitepaper deals with the latter two mitigations – compensation and tolerance. We were motivated by the desire to explore any potential benefit in tailoring the kind of fault tolerance (FT) employed in a computation to the specification of the computation and its goal, instead of either trying to make all intermediate steps be error-free, by error correction (EC), irrespective of their "role"; or by adding redundancy and "voting". There are two aspects to this: (1) Some computations may be *inherently* fault tolerant, may have "built-in" redundancy – they will "succeed" even with a certain amount of internal error – certainly any kind of fault tolerant system can be considered to be redundant, since it, or a part of it, in a fault-free

DOI: 10.4018/978-1-4666-2056-8.ch003

environment does more than it absolutely has to in order to achieve its declared goal; (2) Error correction, though needed, may fail, and we then need to evaluate the degree to which the goal of the computation has been achieved despite the uncorrected error.

In this paper we speculate on how several issues related to FT could be formulated "abstractly", "semantically". While practical application is a theoretical possibility, and a hoped-for long term goal, that is not attempted here. We do not deal at all with issues of computability or complexity. Rather, we wish to open the discussion at a level of generality that captures the concepts, with as few restrictions based on presumed practical limitations as possible. This could eventually lead to posing the "right" questions—for example, questions whose answers might make possible rigorous reasoning about the trade-offs between EC and FT, or about system fault-state behavior. We believe it is interesting to abstract an application domain to a high enough level where we can see what general reasoning can be brought to bear, and where the assumptions need to be strengthened in order to get anything "useful".

Our approach yields some questions in model theory, perhaps interesting in their own right.

However, one major stumbling block on the path from theory to practice is the potential difficulty of even knowing (calculating) if a given property is true in a given model. We also do not attempt to answer questions whether any "real systems" (e.g., biological) fit this model, or under what circumstances this would be a good way to *design* real systems.

We are interested in the possibility of eliminating error correction altogether – and in some cases, also error detection – and simply do "pure fault tolerance" (which we abbreviate to PFT). In other words, if the errors (or "faults" – we don't differentiate between those for this paper) are sparse enough, or weak enough, then we can actually tolerate them, and the computation will "succeed" (to a specified greater or lesser degree), "in the presence" of faults. For example, in the case of a fault tolerant computation, this implies that the fault tolerance will have to be an integral part of the algorithm, tailored for each specification. Another way to state this is that we wish to elevate the fault tolerance to the level of the specification, and not leave it to the "implementation details."

For example, it appears that "standard" quantum algorithms, such as those of Shor and Grover, which rely on quantum error correction for their implementation, would have to be rewritten in a major way in order to incorporate PFT, if that were to be at all possible. For example, there may be a quantum factorization algorithm, different from Shor's, say, which even though it is much less efficient in each run, utilizes true PFT, and therefore, perhaps, the tolerable error threshold will be much higher than the current value, and thus it can actually be built sooner.. In order to approach the conditions for potential use in quantum computation, it seems that certain standard fault tolerance techniques – for example, those involving comparisons of values and voting – cannot be used directly. With a better idea of the potential use of PFT in quantum computation, we could then combine that degree of PFT with EC to get an optimal fault tolerant solution.

We also believe that the so-called "threshold theorems" of quantum computation, which guarantee that error correction will prevail over error proliferation if the error rate is small enough, also have analogs in the PFT paradigm.

The real power of the word "prevail" in the above paragraph is that the *cost* of error correction is vastly overpowered by the *benefit* of the resulting fault tolerant computation. Without taking into account the cost/benefit the problem becomes trivial: simply run a (potentially) infinite number of computations in diagonal fashion and check each "result" as it comes out for correctness. (Apparently, not all quantum algorithms are of this type – in which the answers are efficiently checkable.)