Model Checking and Fault Tolerance

Glenn Bruns and Ian Sutherland

Bell Labs, Lucent Technologies
{grb,ian}@research.bell-labs.com

Abstract. We present an algebraic approach to the model checking of fault-tolerant systems. Fault models and fault-handling mechanisms are modelled using special-purpose process operators. Besides providing for natural models, special-purpose operators allow systems with large state spaces to be verified using systems with small state spaces. To support this verification technique we show that a kind of simulation relation on processes preserves all process operators in tyft/tyxt format.

1 Introduction

Model checking – in which a system model is automatically checked to see if it satisfies a temporal logic formula – has two serious limitations. The size problem is that the state space of the model can grow exponentially with the state space of its components. Additionally, model checking algorithms for many logics have high time complexity. The generality problem is that model checking tells us only that some instance of a system satisfies a property. We usually want to know that a system works properly over a range of parameter values and initial conditions.

These limitations are especially serious in the application of model checking to fault-tolerant systems. By modelling failures of a system component one can increase its possible interactions with other components and thus dramatically increase the state space of the system. Furthermore, one usually wants to show that a fault tolerance mechanism is general purpose; in such cases it is not enough to show that it works for a particular underlying system.

Here we present an approach to the model checking of fault-tolerant systems based on process algebra. To model faults and fault-handling mechanisms we define new, special-purpose process operators. A faulty version of a process is modelled by applying a fault operator to it. For example, suppose process $P$ models a system or system component. To obtain a crash-faulty version of $P$ we define a new operator $Cr$ and apply it to $P$. Similarly, a fault-tolerant or fault-detecting version of a process is defined by applying a process operator to it. This approach has modelling advantages because it is natural to understand faults and fault-handling mechanisms (e.g., triple-modular redundancy [19]) as behavior transformers that are independent of an underlying process or computation.

Defining faults and fault-handling mechanisms as process operators also has technical advantages. We show that if a temporal property holds of a fault-handling mechanism applied to a simple underlying process, then it holds of the mechanism applied to more complex underlying processes. In this way both the
46

size and generality problems of model checking can be reduced. The technique is an application of a general abstraction principle for processes. To apply it here we have had to show that a kind of simulation relation on processes is preserved by every process operator that we might want to define.

Our work uses and extends results on rule formats for structured operational semantics [12, 5]. These results show that certain properties hold of every process built of operators defined in a specific format. The use of special-purpose process operators in specification is discussed by Bloom [4], but not specifically for modelling faults and fault-handling mechanisms. The use of the simulation relation in verification has been explored by Lynch [16], but not for the verification of temporal properties. Our work also draws on the new Process Algebra Compiler (PAC) [8], which takes a process language definition and produces a version of the Concurrency Workbench of North Carolina [9] that can analyze terms of the language.

Our verification approach is property-based: we verify that particular temporal properties hold of a system. This approach is more flexible than that used in [17], in which one shows that a fault tolerance mechanism applied to a faulty system yields behavior equivalent to a nonfaulty system. For example, consider a nuclear reactor protection system, which responds to faults in the reactor by shutting the system down. The protection system ensures safety, but does not ensure the other properties we expect of a normally operating reactor.

The paper is organized as follows. In the next section we show how faults and fault-handling mechanisms can be cleanly modelled using process operators. We then define the simulation relation on processes and show how it helps in verifying properties of complex systems. We illustrate the usefulness of our technique by using it to verify properties of British Rail's slow-scan design. Proofs of the theorems are given in the full version of the paper.

2 Modelling Faults

In hardware fault tolerance, a fault is "a physical defect which may or may not cause a failure" [14]. Software has no faults in this sense. We extend the notion of fault here to include design errors, and consider faults of both hardware and software.

Faults are rarely modelled directly, as that would require their precise nature and location to be known. Instead they are usually modelled by the way they transform the behavior of a system. For example, a crash fault extends the behavior of a system by allowing a crash to occur in any state. In this paper faults are modelled using process operators that conform to the tyft/tyxt rule format [12]. These rules, like the rules of CCS (see Appendix), are used to define the semantics of process operators. We do not have space to present the tyft/tyxt rule format here. However, we should point out that the format is very general: it has been used to describe the operators of CCS, ACP, and other process algebras, as well as the operators used in this paper.

Consider an operator \( Cr \) that models crash faults. A process with crash faults