Model Checking is an automatic verification technique for state-transition systems that are finite-state or that have finite-state abstractions. In the early 1980’s in a series of joint papers with my graduate students E.A. Emerson and A.P. Sistla, we proposed that Model Checking could be used for verifying concurrent systems and gave algorithms for this purpose. At roughly the same time, Joseph Sifakis and his student J.P. Queille at the University of Grenoble independently developed a similar technique. Model Checking has been used successfully to reason about computer hardware and communication protocols and is beginning to be used for verifying computer software. By expressing a system’s specifications in temporal logic, the Model Checker can perform an exhaustive search to determine if the specification is true. In those cases where the specification does not hold, the Model Checker produces a counterexample execution trace. We have found this feature to be extremely useful for finding obscure errors in complex systems. However, Model Checking is vulnerable to the state-explosion problem, which can occur if the system being verified has many processes that execute in parallel or complex data structures. In some cases, the state-explosion problem is inevitable, but over the past 27 years we have made considerable progress in overcoming this problem for certain classes of state-transition systems that occur often in practice. In this talk, I will describe what Model Checking is, how it works, and the main techniques that have been developed for combating the state-explosion problem.