Dynamic Invariant Detection for Program Understanding and Reliability
MIT 2001-01

Progress Report: July 1, 2002—December 31, 2002

Michael Ernst

Project Overview

The Program Analysis Group (PAG) of MIT is collaborating with Tadashi Araragi of NTT on problems raised by NTT's Erdös internet agent system. The goal is to formally verify mobile agent systems in which agents may be widely separated and may move about a network, yet must communicate, coordinate, and negotiate in order to perform useful work. Examples include internet marketplaces (in which directories of shops must be maintained), auctions (in which programs may be dynamically downloaded), and airline ticket reservation (in which multiple ticket agencies must have a consistent view of a database). In each of these cases, it is crucial that the system perform correctly, yet it is difficult to understand and verify it. PAG has provided several technologies that can increase confidence in such systems.

Progress Through December 2002

One of PAG's key contributions is a technique for automatically generating operational abstractions, which are formal mathematical descriptions of program operation. They are syntactically identical to program specifications, but they describe what the program does (as opposed to what it is supposed to do), which makes them uniquely useful for many program analysis and understanding tasks. A description of program behavior is required for most program understanding, maintenance, and verification tasks. Specifications are rarely present, but operational abstractions can be automatically generated by PAG's Daikon system, which requires only a program and a test suite.

Between July and December 2002, we have made additional progress on the project's goals. Infrastructure is now largely in place, though we have continued to support users, for instance by making 9 public releases of the Daikon tool during the period and by adding new functionality requested by users. Our research has focused on two main areas of interest to NTT: automated theorem-proving and detection of temporal properties. The first is directly applicable to verification goals, and the second represents a domain of interest to for the Erdos system.

Because the final goal of this NTT-MIT project is program verification, PAG has continued to press forward on proofs of program correctness. Our focus during this period of the grant has been on automating proofs. (During this period of the grant, we switched from the LP to the Isabelle proof assistant, as we had planned in our last progress report.) Generally speaking, there are two sorts of theorem proving tools: automatic theorem provers and proof assistants. The first variety, automatic theorem provers, are easy to use, but relatively weak. No user interaction is permitted, and they have limited built-in ability to perform search, so they can handle proofs of only relatively simple properties. This makes them useful in certain well-understood circumstances that can be tailored to their abilities, but they scale poorly to new situations. The second variety, proof assistants, provide no more search capabilities, but they do present a user interface that permits users to specify each step of a proof. The proof assistant checks the steps and performs low-level bookkeeping. The tools are applicable to almost any situation that a human can imagine (and can structure a proof for). The problem with such tools is their lack of automation: humans must make all decisions about the proofs. Since human expertise is the most expensive

part of most systems, this approach does not scale. Our goal is to provide the best of both worlds: the automation of an automatic theorem prover and the power of a proof assistant. We have made substantial steps toward this goal. A proof assistant requires two types of input: lemmas to be proved, and the tactics (such as case analysis, induction, deduction, or the like) that are used to prove them. For a collection of small but realistic distributed algorithms, the Daikon tool has automatically proposed nearly all the necessary lemmas, and another tool that we have built has proposed nearly all the required tactics. The result is a reduction of over 90% in the human effort required for theorem-proving. The system has also led to a new, simpler proof for one of the algorithms.

Our other focus has been on detecting temporal properties (invariants), which are of particular interest to NTT and which are crucial in understanding and verifying distributed systems such as the Erdos distributed agent system. As planned in our previous progress report, we have implemented a technique for detecting such properties. In fact, we have devised and implemented three distinct techniques, each with its own strengths and weaknesses. We briefly discuss them here.

The first technique determines which calling sequences are consistent with automatically generated program specifications (operational abstractions). In particular, the Daikon tool generates an operational abstraction for each procedure in an interface. Then we automatically perform logical analysis, using a specialized theorem-prover, to determine which preconditions are compatible with which postconditions. When the properties are incompatible, then it is not permitted to call one routine immediately after the other. Experiments show that this technique is more accurate than methods proposed by other researchers, and that it converges to a good answer more quickly and with a smaller test suite.

The second technique is a general-purpose dynamic detector of temporal invariants. The invariants are built up out of the basic elements (exists, always, before, after) that make up temporal logics. Dynamic instantiation prevents the need to know all events before processing starts, and optimizations permit retraction of events that are later determined not to be true.

The third technique is an optimized dynamic detector of temporal invariants that detects a smaller grammar of properties, but is able to operate on much larger traces. It also incorporates certain universal and existential quantifiers.

We have run all three techniques on traces from Java libraries, from a commercial application, and from Erdos programs. We have discovered problems with the libraries, have verified properties of the application test suites, and have recovered desired properties of the Erdos programs. At present our greatest need from NTT is feedback about what extensions would be most useful, and additional Erdos programs to use in further experiments.

Research Plan for the Next Six Months

Having obtained good research results in our laboratory, we wish to disseminate and use these at NTT and elsewhere. We plan to generalize the theorem-proving results to additional application domains and to automate the parts of proofs that still require some human interaction. Our examinations of the proofs have indicated promising directions. With respect to temporal invariant detection, we are pleased with our current results, but we wish our future efforts to be guided by industrial applications. We are looking forward to receiving additional programs that will indicate either that we have been successful or the directions in which we should proceed next. In their absence, we will continue with our experiments on Java libraries and commercial applications.