Porting IOMMU to mCertiKOS, a Verified Operating System

Jacob Geiger
Advised by Zhong Shao
Project Proposal for CPSC 490
4 February 2015

Introduction

In formal verification, the source code of a piece of software is accompanied by a mathematical proof of its correctness. Formal verification offers a checkable guarantee that a given piece of software conforms to a stated specification; with a sufficiently descriptive specification, formal verification can prove a given piece of software to be “bug free.” In practice, formally verified software is far more labor-intensive to write than software developed using conventional testing strategies. Nevertheless, in practice, formal verification has been used to produce a certified C compiler, and it provides an attractive option for situations which require a critical level of confidence in software.

This project aims to increase the functionality of mCertiKOS, a verified operating system microkernel under development. In particular, porting IOMMU (input-output memory management unit) to mCertiKOS would permit it to use a standard interface specified by Intel to interact with external devices by mapping the devices’ virtual memory into the OS’s, and provide a direct way for virtual machines running on top of mCertiKOS to access these devices. Verifying IOMMU poses some difficulty due to its reliance on devices external to mCertiKOS itself.

Background

Coq
Coq is a programming language with a number of built-in constructs that allow the user to write and automatically verify proofs. It additionally provides utilities for automating certain aspects of proofs, making it possible to verify proofs that would be too tedious or unfeasibly long to
write out by hand. It is possible in Coq to model a different programming language—in the case of mCertiKOS, the C programming language—and to write out its semantics. With this information, it is possible then to translate programs from C into Coq’s model of C, and to prove assertions about those programs. Verification of mCertiKOS is accomplished using this method.

To verify mCertiKOS in Coq, the code of mCertiKOS is divided into sections, each of which is divided into interdependent “layers.” The most basic features are present in the lower layers; these combine to form more complicated features that are built on top of previous layers. For each layer, a “deep specification” is written containing all assertions about that particular section of code that are meant to be exposed to other sections of mCertiKOS. That is, it should be possible to learn everything relevant about the functionality of a particular piece of code by reading its deep specification. The deep specification is then proved in Coq. This deep specification can then be used in verifying higher layers.

**Clight/CompCert/Clightgen**
Before verification, mCertiKOS is written in C. However, certain aspects of C, such as side effects during expression evaluation, substantially complicate proof writing. Clight is a subset of C modeled in Coq that excludes these undesirable properties. CompCert is a verified C compiler; the utility Clightgen uses CompCert to translate unverified C source into Clight, permitting formal verification in Coq.

**mCertiKOS/VeriKOS**

mCertiKOS is a operating system microkernel; it provides a minimal amount of functionality necessary for an operating system, preferring to offer other functionality through processes running in user space. Through its support of virtualization, mCertiKOS allows unverified software to run in virtual machines on top of this verified microkernel. By segregating resource use among these virtual machines, mCertiKOS can maintain security even when running untrusted software.

VeriKOS is an unverified operating system designed similarly to mCertiKOS; the process of developing mCertiKOS generally involves porting features from VeriKOS into mCertiKOS and verifying them. In particular, the target of this project—IOMMU—is in the process of being ported from VeriKOS to CertiKOS.

**IOMMU**

IOMMU maps the physical address space on the computer to the virtual address space of external devices attached to it; this address space can then be used for input/output from and onto...
these devices. It also provides ways for external devices to communicate interrupts. IOMMU is designed to support virtualization, providing virtual machines efficient access to these devices. This aspect is critical for mCertiKOS’s use of virtual machines to segregate untrusted software.

**Project Description**

The process of porting IOMMU from VeriKOS to mCertiKOS has been divided into five parts, each encompassing the porting of a constituent driver:

1. ACPI (Advanced Configuration and Power Interface), which abstracts the hardware of the device
2. APIC (Advanced Programmable Interrupt Controller)
3. MSI (Messaged Signal Interrupts)
4. PCI (Peripheral Component Interconnect)
5. Intel VTd, which is Intel’s standard for supporting virtualization in device I/O

So far, ACPI, APIC, and MSI have already been ported to mCertiKOS as unverified C code. It remains to port PCI and Intel VTd as unverified C code, then to verify each of these drivers in order. Before PCI and Intel VTd are ported, their functionality must be broken into layers to aid in proof writing, as described in the previous section. The verification may pose some difficulty due to its reliance on external hardware.

**Deliverables**

The deliverables for this project include:

1. C source code for porting PCI and Intel VTd from VeriKOS to mCertiKOS
2. Layers for PCI and Intel VTd
3. Coq deep specifications and proofs for these drivers
4. A written report explaining the work done, and summarizing theorems proved using Coq
5. An oral presentation to the Department of Mathematics in accordance with the requirements of the joint Computer Science & Mathematics major