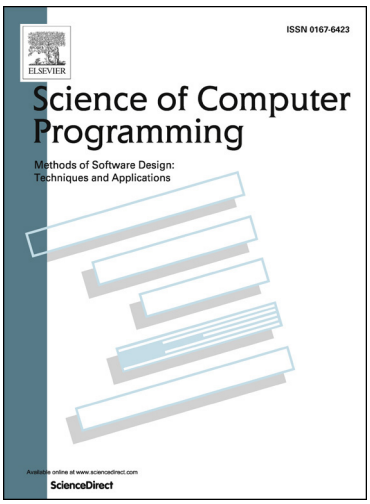
Accepted Manuscript



Formal proof of dynamic memory isolation based on MMU

Narjes Jomaa, David Nowak, Gilles Grimaud, Samuel Hym

PII: S0167-6423(17)30133-8
DOI: http://dx.doi.org/10.1016/j.scico.2017.06.012
Reference: SCICO 2116

To appear in: *Science of Computer Programming*

Received date: 2 January 2017
Revised date: 22 June 2017
Accepted date: 25 June 2017

Please cite this article in press as: N. Jomaa et al., Formal proof of dynamic memory isolation based on MMU, *Sci. Comput. Program.* (2017), http://dx.doi.org/10.1016/j.scico.2017.06.012

Formal Proof of Dynamic Memory Isolation Based on MMU[☆,☆☆]

Narjes Jomaa, David Nowak, Gilles Grimaud, Samuel Hym

CRIStAL, CNRS & Lille 1 University, France

Abstract

For security and safety reasons, it is essential to ensure memory isolation between processes. The memory manager is thus a critical part of the kernel of an operating system. It is common for kernels to ensure memory isolation through a piece of hardware called memory management unit (MMU). However an MMU by itself does not provide memory isolation. It is only a tool the kernel can use to ensure this property. In this paper we show how a proof assistant such as Coq can be used to model a hardware architecture with an MMU, and an abstract model of microkernel supporting preemptive scheduling and memory management. We proceed by making formally explicit the consistency properties that must be preserved in order for memory isolation to be preserved.

Keywords: Formal proof, Memory isolation, Microkernel, Coq.

1. Introduction

Modern operating-system kernels make it possible to share computer resources between untrusted processes, and to rapidly deal with external events, e.g., arrival of a network packet that would be lost if not dealt with immediately. In this context, both for safety and security reasons, it is important to respectively prevent accidental and malevolent access by a process to an address outside its own address space. On modern computers, kernels ensure memory isolation with the help of a piece of hardware called memory management unit (MMU). An MMU is a hardware component through which all memory accesses must go. It translates a virtual memory address to a physical address if there is indeed a corresponding one in the current setting. It also checks whether in the current setting accessing this address is allowed. It is indeed a common design to have the kernel space always mapped for efficiency reasons but not accessible while in user mode. For this to work properly, the kernel has to maintain page

[☆]This work was partially supported by the Celtic-Plus Project ODSI C2014/2-12, CNRS Action Spécifique Sécurité, and IRCICA USR 3380.

[☆☆]A preliminary version of this work appeared in the proceedings of the 10th International Symposium on Theoretical Aspects of Software Engineering (TASE 2016) [1].