



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Proving invariants in the deep embedding</b>	<b>2</b>
2.1	Preliminary experiments . . . . .	2
2.2	Modelling the PIP state in the deep embedding . . . . .	3
2.3	1 <sup>st</sup> invariant and proof . . . . .	3
2.3.1	Invariant in the shallow embedding . . . . .	3
2.3.2	Modelling the <i>getFstShadow</i> function . . . . .	5
2.3.3	Invariant in the deep embedding . . . . .	11
2.3.4	Invariant proof . . . . .	12
2.3.5	The Apply approach . . . . .	16
2.4	2 <sup>nd</sup> invariant and proof . . . . .	17
2.5	3 <sup>rd</sup> invariant and proof . . . . .	19
2.6	Observations . . . . .	23
2.6.1	Deep vs shallow . . . . .	23
2.6.2	Importance of Hoare triple rules . . . . .	23
2.6.3	Lack of a Pattern matching Construct . . . . .	24
2.6.4	Dealing with values in the deep embedding . . . . .	24
2.6.5	Defining Shallow functions . . . . .	25
2.6.6	Deep implementation of shallow functions . . . . .	25

# List of Scripts

2.1	PIP state in the deep embedding . . . . .	3
2.2	getFstShadow invariant in the shallow embedding . . . . .	4
2.3	getFstShadow function in the shallow embedding . . . . .	4
2.4	nextEntryIsPP property . . . . .	5
2.5	partitionDescriptorEntry property . . . . .	5
2.6	getShlidx definition in the deep embedding . . . . .	6
2.7	Index successor function in the shallow embedding . . . . .	6
2.8	Rewritten shallow index successor function . . . . .	6
2.9	Definition of Succ . . . . .	7
2.10	Definition of SuccD . . . . .	7
2.11	Functions called in SuccD . . . . .	8
2.12	Definition of PlusR . . . . .	9
2.13	Definition of SuccRec . . . . .	9
2.14	readPhysical function in the shallow embedding . . . . .	9
2.15	Rewritten shallow readPhysical function . . . . .	10
2.16	Definition of ReadPhysical . . . . .	10
2.17	Definition of getFstShadowBind . . . . .	10
2.18	Rewritten nextEntryIsPP property . . . . .	11
2.19	getFstShadow invariant definition . . . . .	11
2.20	proof of the getFstShadow invariant . . . . .	12
2.21	getShlidxWp lemma definition and proof . . . . .	13
2.22	succWp lemma definition and proof . . . . .	14
2.23	succW Lemma definition and proof . . . . .	15
2.24	readPhysicalW Lemma definition and proof . . . . .	16
2.25	Lifting Succ and readPhysical to quasi-functions . . . . .	16
2.26	getFstShadowApply definition . . . . .	17
2.27	writeVirtual function in the shallow embedding . . . . .	17
2.28	Rewritten shallow writeVirtual function . . . . .	17
2.29	WriteVirtual definition . . . . .	18
2.30	writeVirtualInvNewProp invariant definition . . . . .	18
2.31	writeVirtualWp lemma definition and proof . . . . .	19

2.32	initVAddrTable in the shallow embedding . . . . .	20
2.33	Maximum index . . . . .	20
2.34	LtLtb definition . . . . .	21
2.35	writeVirtual new definition . . . . .	21
2.36	ExtractIndex definition . . . . .	21
2.37	initVAddrTable definition in the deep embedding . . . . .	22
2.38	initVAddrTableNewProp invariant in the deep embedding	22
2.39	succWp false lemma definition . . . . .	24

# List of Figures

# Acronyms

**API** Application Programming Interface.

**DSL** Domain Specific Language.

**HAL** Hardware Abstraction Layer.

**IAL** Interrupt Abstraction Layer.

**IPC** Inter-Process Communication.

**MAL** Memory Abstraction Layer.

**MMU** Memory Management Unit.

**OS** Operating System.

**SOS** Structural Operational Semantics.

**TCB** Trusted Computing Base.

# 1. Introduction

The following report describes the activities carried out during a 12-week, full-time internship at the Research Center in Computer Science, Signal and Automatic Control of Lille (CRIS<sup>t</sup>AL). This internship revolves around the PIP pretokernel and DEC currently being developed by the 2XS team. PIP is a minimal OS kernel with provable memory isolation using Hoare logic. The deep embedding or DEC, as opposed to the shallow embedding, is an intermediate language for the translation of PIP to C. Specifying programs in the deep embedding has the great advantage of simplifying their syntactic manipulation. It also ensures a stricter structuring of program expressions. **Therefore, we want to compare deep and shallow proofs and, more precisely, check whether this structure is reflected in the proofs done in the deep embedding ?**

To that end, we will model three different functions in the deep embedding. The first one reads the memory. The second one writes in the memory. The last one is a recursive function. Then, we will prove invariants about these functions. One of the invariants propagates all of PIP's properties which include memory isolation, vertical sharing, kernel data isolation and consistency. We chose a modular approach in our implementation and we engineered our proofs correspondingly. We also did some preliminary work mostly to get familiar with Hoare logic and the deep embedding.

The first part of the report offers an overview of the CRIS<sup>t</sup>AL laboratory, the 2XS team and their research activities. The second part is dedicated to the PIP protokernel and focuses on its proof oriented design, its properties, its data structures, Hoare logic theory and, more importantly, the deep embedding and its constructs. The last part is dedicated to our contributions, detailing how we modelled the required functions, how we engineered our proofs in the deep embedding and our observations about the comparison between the deep and shallow proofs.