Addressing the Performance Bottlenecks of Device Driver Synthesis

Alexander Legg

Supervised by
Leonid Ryzhyk, Nina Narodytska, and Gernot Heiser

August 13, 2015

Contents

Abstract

Device drivers are notorious for their contribution to the failure of software systems. A driver forms the interface between two complicated systems: an operating system and a hardware device. As a result, driver development is complicated and highly prone to human error. Driver reliability has been the focus of much research over the past two decades. One result of this research is device driver synthesis.

Device driver synthesis aims to remove human error from the development process by removing humans. A synthesis tool can take a formal specification of requirements and produce a system. Thus, a driver can be synthesised by providing the tool with a specification of the OS and the hardware that it must interface with. The resulting driver is guaranteed to be correct with respect to the specifications.

The drawback of this approach is that synthesis is a computationally intensive task belonging to 2-EXPTIME. In practice synthesis tools are capable of producing drivers though there remain cases where synthesis is impossible. In this thesis I will present an algorithm that provides results on many of these previously unattainable cases.

Acknowledgements

Acknowledge some people

Chapter 1

Introduction

This is the introduction.

1.1 Device Drivers

Explain what a driver is (just in case.)

Drivers are bad. Talk about the linux kernel bugs paper. Mention the drawbacks of their approach.

Consequences of buggy drivers

Will talk about other approaches to the problem in the next chapter

1.2 Synthesis

Explain what synthesis is

Briefly describe the current state of the art

Explain state space explosion

1.3 Synthesis with SAT

Explain SAT solvers

Explain how they might help with state space explosion

(Explain that there is room for both approaches in the world)

Explain QBF and/or why this is nontrivial

Chapter 2

Related Work

Chapter 3

Formalisation of Software

Synthesis is a process that demands mathematical formalisation in order to provide a strong guarantee of the correctness of the resultant software. As such we require a mathematical language to describe the system we wish to produce, the environment in which it operates, and the properties we want the system to adhere to. This chapter will outline that language and the ways we can reason about what we describe in it.

3.1 Reactive Systems

Devices drivers are an example of a reactive system. A reactive system is a system that is in a continuous process of responding to input from its environment. A driver accepts requests from the operating system and state information from the device, and it responds by sending commands to the device and reporting to the operating system.

Büchi automata [1] are a formalisation of reactive systems. Büchi automata are ω-automata, which means they describe finite systems that accept an infinite stream of input. This is a useful formalism for synthesis because we want to create finite systems that responds to input continuously or infinitely.

Like all ω-automata, the language of a Büchi automaton is ω-regular. A regular language over the alphabet Σ is

- The empty language , *or*
- A singleton language $\{a\}$ for $a \in \Sigma$, *or*
- For two regular languages A and B:

- $A \cup B$ the union of those languages, *or*
- $A \,.\, B$ the concatenation of those languages, *or*
- $A*$ the Kleene operation on that language.

An ω-regular language is a regular language with infinitely long words.

3.2 Temporal Logic

3.2.1 LTL and CTL

3.3 Reactive Synthesis

3.3.1 Synthesis as a Game

3.3.2 GR(1) Games

3.4 Binary Decision Diagrams

3.5 Abstraction

3.6 SAT

3.7 Interpolation

Chapter 4

Synthesis with SAT

Chapter 5

Evaluation

Chapter 6

Conclusion

Bibliography

[1] J. Richard Büchi. On a decision method in restricted second order arithmetic. In *Logic, Methodology and Philosophy of Science (Proc. 1960 Internat. Congr .)*, pages 1–11. Stanford Univ. Press, Stanford, Calif., 1962.