# Realizing Iterative-Relaxed Scheduler in Kernel Space

Master-Arbeit
Sreeram Sadasivam
2662284

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik

Fachgebiet Dependable Embedded
Systems and Software
Prof. Neeraj Suri Ph.D

Realizing Iterative-Relaxed Scheduler in Kernel Space
Master-Arbeit
2662284

Eingereicht von Sreeram Sadasivam
Tag der Einreichung: 16. Marz 2018

Gutachter: Prof. Neeraj Suri Ph.D
Betreuer: Patrick Metzler

Technische Universität Darmstadt
Fachbereich Informatik

Fachgebiet Dependable Embedded Systems and Software
Prof. Neeraj Suri Ph.D

## Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Master-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die schriftliche Fassung stimmt mit der elektronischen Fassung überein.

Darmstadt, den 16. Marz 2018 Sreeram Sadasivam

Contents

1 Background 1
1.1 Software Verification . . . . . 1
1.2 Multithreaded Programming . . . . . 1
1.3 Model Checking . . . . . 2
1.4 Symbolic Execution . . . . . 2
1.5 Iterative Relaxed Scheduling . . . . . 2
1.6 Deterministic Multi-Threading . . . . . 2

Bibliography 2

# List of Figures

# List of Tables

## Abstract

Abstract comes here...

# 1 Background

## 1.1 Software Verification

Software programs are becoming increasingly complex. With the rise in complexity and technological advancements, components within a software have become susceptible to various erroneous conditions. Software verification have been perceived as a solution for the problems arising in the software development cycle. Software verification is primarily verifying if the specifications are met by the software[1].

There are two fundamental approaches used in software verification - dynamic and static software verification[1]. Dynamic software verification is performed in conjunction with the execution of the software. In this approach, the behavior of the execution program is checked- commonly known as Test phase. Verification is succeeding phase also known as Review phase. In dynamic verification, the verification adheres the concept of test and experimentation. The verification process handles the test and behavior of the program under different execution conditions. Static software verification is the complete opposite of the previous approach. The verification process is handled by checking the source code of the program before its execution. Static code analysis is one such technique which uses a similar approach.

The verification of software can also be classified in perspective of automation - manual verification and automated verification. In manual verification, a reviewer manually verifies the software. Whereas in the latter approach, a script or a framework performs verification.

Software verification is a very broad area of research. This thesis work is focussed on automated software verification for multithreaded programming.

## 1.2 Multithreaded Programming

Computing power has grown over the years. Advancements are made in the domain of computer architecture by moving the computing power from single-core to multi-core architecture. With such advancement, there were needs to adapt the programming designs from a serialized execution to more parallelizable execution. Various parallel programming models were perceived to accommodate the perceived progression. Multithreaded programming model was one of the designs considered for the performance boost in computing.

Threads are a small tasks executed by a scheduler on an operating system, where the resources such as the processor, TLB (Translation Lookaside Buffer), cache, etc., are shared between them. Threads share the same address space and resources. Multithreading addresses the concept of using multiple threads for having concurrent execution of a program on a single or multi-core architectures. Inter-thread communication is achieved by shared memory. Mapping the threads to the processor core is done by the operating system scheduler. Multithreading is only supported in operating systems which has multitasking feature.

Advantages of using multithreading include:

- Fast Execution
- Better system utilization
- Simplified sharing and communication

- Improved responsiveness
- Parallelization

Disadvantages:

- Race conditions
- Deadlocks with improper use of locks/synchronization
- Cache misses when sharing memory

### Partial Order Reduction

### Lipton

### Dynamic POR

## 1.3 Model Checking

## 1.4 Symbolic Execution

## 1.5 Iterative Relaxed Scheduling

## 1.6 Deterministic Multi-Threading

## Bibliography

[1] Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. Fundamentals of software engineering. Prentice Hall PTR, 2002.