# Executive Summary

## YAGAL: Yet Another GPGPU Abstraction Library

This master thesis documents the development of a *C++* GPGPU framework. This framework is named *YAGAL*, and it stands for Yet Another GPGPU Abstraction Library. *YAGAL* is an attempt at constructing a framework, that utilizes the *CUDA Driver API* for device management and *LLVM* for code generation at runtime, to be compiler independent without relying on *OpenCL*. *YAGAL* provides abstractions for the developer, without knowledge of the GPGPU programming model, to describe logic through *YAGAL* Vectors for memory management, and *YAGAL* Actions for kernel building blocks.

The initial thesis work revolves around investigating existing frameworks. These frameworks serve as a baseline for our understanding of the problem domain, and an inspiration source for our design choices. These frameworks are presented as related works, which include well established frameworks and ongoing research projects. The related works consist of *Thrust*, *C++ AMP*, *Bolt*, *SkelCL*, and *PACXX*.

The design of *YAGAL* is done by first identifying design principles for API design, and then designing the API and architecture separately. The API is designed first, covering data types and functions, by discussing possible approaches. The architecture is then designed to provide the features needed by the API, and introduces the technologies used by *YAGAL*, namely *LLVM*, the *CUDA Driver API*, and *PTX*.

The implementation documentation of *YAGAL* covers the process from design to implementation, and highlights the technically interesting parts. It walks through the development in incremental steps, starting with management of *CUDA* device and memory, execution model, the action abstraction, the utilization of *LLVM Intermediate Representation* and the features it provides us with, how threads are organized according to the amount of vector elements, and how we enable the use of *PTX* for the developer.

During the implementation of *YAGAL*, technical challenges were discovered. The first challenge is, that implementation of anonymous functions in a framework that does not build on *OpenCL* and that does not provide a compiler is technically difficult, and none of the related works has an implementation that does this. The second challenge is, that the inclusion of *LLVM* in *YAGAL* severely impacts compilation time, and the development-debug loop. We discuss the reason and impact of these challenges, and propose approaches to how they can be addressed.

An evaluation and comparison methodology is defined and used to evaluate *YAGAL* and compare it to the related works. The methodology is split into two parts; a general comparison, which consists of measurable metrics, and a usability evaluation based on Cognitive Dimensions of Notations.

We discover that *YAGAL* is slower than *Thrust* while being more sparse on features. This means that there is no incentive to use *YAGAL* instead of *Thrust* in regards to performance. In terms of usability, the abstractions of *YAGAL* are simple mental models for building kernels, and this might be worth investigating further, possibly in other less performance oriented contexts.

We finally conclude that development of a GPGPU framework using the *CUDA Driver API* while being compiler-independant is indeed possible, but when high level abstraction is needed, this design impacts the required amount of implementation work in a negative direction compared to other designs.