**Parallel Programming Technology Lab: Proposal**

**Background:**

Every program contains data dependencies. A data dependency means that the value of some variable depends on another variable. There are two types of analyses of the data dependencies of a program: static, which means the analysis of the program before its execution, and dynamic, which means the analysis at compile time. We will use three methods of static analysis: Use-Def-Chain, Alias Analysis and Interprocedural Analysis. Use-Def-Chain describes

LLVM is an compiler architecture which offers many tools to analyze program code. It can be used to apply data dependency extraction to programs.

**Motivation:**

Being able to extract these data dependencies automatically from a program is extremely helpful in terms of runtime of the program. Analyzing data dependencies can lead to a significant speedup of the program.

**Objectives:**

In this project, we will create a program that extracts data dependencies from a program. The result will be converted to the DiscoPoP format and then be compared to the static method PLUTO and the dynamic method DiscoPoP. The comparison will give insights to how important the static data dependency analysis is, compared to the dynamic one.

**Solution:**

Our software will use LLVM. Our software will be a composition of different existing analysis methods including Use-Def-Chain, Alias Analysis and Interprocedural Analysis. The conversion of the result to the DiscoPoP format will be done with an existing Python tool. In the end, there will be three different diagrams containing the data dependencies of one program, by using three different extraction methods. As test programs we will use benchmark suits like Polybench, NPB and BOTS.

**Expected Results:**

**Work Plan:**

**Responsibilities:**

**Time line:**

**References:**

<https://de.wikipedia.org/wiki/LLVM>

<https://en.wikipedia.org/wiki/Use-define_chain>

<https://en.wikipedia.org/wiki/Alias_analysis>

<https://en.wikipedia.org/wiki/Interprocedural_optimization>

**OLD:**

Extracting data dependencies from a program is crucial for analyzing and optimizing the program in means of complexity and runtime. A data dependency means that some variable has a value which is earlier defined by another variable. Finding such dependencies can help to speed up a program e.g. by replacing a variable which has a constant value by the value itself. There are multiple methods to find data dependencies, such as USE-DEF-analysis, alias-analysis and inter-procedure analysis. In this project, we will combine these analyses to extract data dependencies from programs using LLVM, then convert the found data dependencies to the DiscoPoP format and compare the results with a dynamic technique called “DiscoPoP” and another static technique called “PLUTO”. As test programs where to extract the data dependencies from, we will use benchmark suits like Polybench, NPB and BOTS.

Problem:

Finding data dependencies to analyze programming code.

Aim:

Then compare static extracting data dependencies techniques with dynamic extracting data dependencies technique DiscoPop and static technique PLUTO.

Techniques:

Use-Def analysis, alias, inter-procedure, etc.

Process:

Implement static extracting data dependencies techniques, execute them on benchmark suits and convert results to DiscoPoP format, so that they can be compared with other techniques.

Tools:

LLVM tool which is used to execute our software on Polybench, NPB and BOTS benchmark suits and extract data dependencies.