Optimizing Kater's Inclusion Algorithm

One way to construct secure systems is to design programs that are correct by construction (verified). The first step in doing so is to formally specify the behaviors of a program. In the context of concurrency, programmers often assume that the behaviors of a multithreaded program can be expressed as interleavings of its threads. This assumption, however, is incorrect, as compiler and/or hardware optimizations generate additional "weak" behaviors that cannot be expressed as interleavings. The formal models describing the exact behaviors that concurrent program can exhibit are called *weak memory models*.

But how can we reason about different memory models, for example, to prove the correctness of program transformations or to prove that a program will also behave correctly under a different memory model? Kater is a state-of-the-art tool that can prove such metatheoretic queries completely automatically, by translating them to a language-inclusion problem between regular languages. Crucially, the performance of Kater depends on the effectiveness of its inclusion algorithm.

The goal of this thesis is to optimize Kater's inclusion procedure. As a first step, instead of first saturating the NFAs involved and then running the inclusion algorithm, we will perform saturations lazily, as part of the inclusion procedure. Then, we will improve the inclusion algorithm itself, by employing an antichain-based optimization.

Goals:

- Lazily perform NFA saturations
- Benchmark the performance difference

Extension Goals:

- Investigate further optimizations