Institut für Systemsoftware

O.Univ.Prof. Dr. Hanspeter Mössenböck

JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Safe Concurrency and Parallelism with Channels and Transactions with a Sequential Programming Style

Master thesis for …
Matr.Nr.: …
Email: …@...

Concurrent and Parallel Programming are intrinsically difficult. Instead of reasoning about a single sequence of operations, one has to reason about multiple and all possible interleavings. Unfortunately, this is not all. Most languages provide plenty of abstractions to solve the various common problems. However, these abstractions are rarely designed to be used in combination, which results in race conditions, deadlocks, and other forms of unexpected behavior that causes bugs. To avoid such accidental complexity, we argue that such programming abstractions need to be carefully designed to work together [1].

In this thesis, we want to explore how Communicating Sequential Processes, known for instance from the Go language's channels, transactional memory, Erlang-style actors, and Clojure-style promises could be designed in a way that we can use them in combination without causing accidental complexity. These abstractions share "blocking" as their core property. For some, the resulting programming model is a natural extension to sequential programming, and thus, we want to see how the different abstractions can be safely combined to solve a wide range of different concurrency and parallelism issues.

The scope of this thesis is as follows:

- Implement abstractions such as channels, communicating process, Erlang-like actors, and transactional memory domains for SOM_{NS}, a dynamic language for concurrency research based on the Truffle framework [2, 3].
- Adapt the concurrency abstractions so that their combined use does not introduce unexpected concurrency issues, while preserving their usefulness for their initial use cases.

The work's progress should be discussed with the supervisor at least every 2 weeks. Please note the guidelines of the Institute for System Software when preparing the written thesis.

Supervisor: Dr. Stefan Marr

[1] Why is Concurrent Programming Hard? http://stefan-marr.de/2014/07/why-is-concurrent-programming-hard/

[2] http://ssw.uni-linz.ac.at/Research/Projects/JVM/Truffle.html

[3] Würthinger, T., Wöß, A., Stadler, L., Duboscq, G., Simon D. and Wimmer, C. "Self-Optimizing AST Interpreters." In Proc. of the 8th Dynamic Languages Symposium, 2012.