MSc Thesis: A Type System for Ensuring Safe Structured Concurrency in Scala

Student: Sebastian Willenbrink (stwi@kth.se)
Supervisor: Associate Professor Philipp Haller, EECS/TCS

Background

Concurrent programming is known to be difficult due to various hazards including data races, deadlocks, etc. In the area of programming languages, a major focus has been to design programming languages and systems that ensure the safety of concurrent/parallel code without affecting performance negatively or restricting the flexibility of programmers. A successful example of a recent language that ensures data-race freedom at compile time is Rust. Rust's approach relies on a sophisticated type system based on ownership and borrowing, concepts that have been studied during the last 25 years.

The high-level goal of this thesis project is to bring some of the ideas behind Rust and related languages to a language that focuses more on functional programming, in this case Scala. The proposed approach is based on the Scala extension LaCasa [1], a type system with a notion of isolation and uniqueness which prevents aliasing. By extending LaCasa with threads that are isolated from each other we can guarantee data-race freedom.

Recently, there have been many different approaches to safe structured concurrency. The LaCasa paper itself includes an extension for actor-based concurrency [1]. In [2], a lattice based approach is used to guarantee deterministic concurrency. In “Habanero-Java: the new adventures of old X10” [3] the Async-Finish Model (AFM) is described, a variant of the Fork-Join Model familiar from e.g. C. The AFM has many desirable properties which this thesis project attempts to combine it with the isolation guarantees from LaCasa to create a type system for safe structured concurrency.

Research Question

The research question can be split into multiple sub-questions:

> Can the existing type system of Scala be extended to support safe structured concurrency where data races are prevented at compile time? Is it possible to establish important properties including type preservation and progress? Is this system deterministic and guaranteed to be deadlock free? How easy is it write parallel programs using this extension?

Hypothesis

The hypothesis the project tries to evaluate is the following:

> It is possible to extend Scala's existing type system with safe structured concurrency in a way that ensures data-race freedom. By leveraging existing concepts, such as object capabilities and isolation, the proofs for type preservation, progress, determinism and

deadlock freedom are feasible. Writing parallel programs requires only a low overhead by making use of the AFM.

Research Method

1. Read up on related work, primarily LaCasa.
2. Devise a type system extending LaCasa following the AFM such that parallel tasks are isolated from each other to ensure data-race safety by preventing access to shared mutable state.
3. Formalize the main ideas of the type system in the context of a small-step operational semantics for a small object-oriented core language.
4. Prove important properties like progress and preservation for this small core language. This proof will be done with pen-and-paper.
5. Prove determinism and deadlock freedom for this small core language.
6. Demonstrate the flexibility and annotation overhead of the type system on a range of task-parallel programs written using Async and Finish. This demonstration will be based on theoretical examples as no prototype exists.

Background of the Student:

I have taken multiple courses in the area of formal methods and programming languages, with the two most relevant being a course on programming language semantics [5] and a seminar on type systems [6]. Through these I learned how type systems and semantics are formalized and familiarized myself with concepts like simply typed lambda calculus and dependent types.

Suggested Examiner at KTH:

Suggested Supervisor at KTH:

Associate Professor Philipp Haller
I have been in contact with Philipp Haller and he has expressed interest in serving as supervisor.

Resources:

Philipp Haller is familiar with the topic. Besides that, I am not aware of any significant resources existing at the department.

Eligibility:

I have completed more than 60 ECTS on an advanced level. All the courses relevant to the thesis were completed at the TUM. I've completed the course on theory of knowledge and research methodology (DA2210).

Study Planning:

I have received grades on all courses except DH2323 as I have not yet finished the practical project of the course. Most of the work is done and only the report is missing.

References:

[1]: LaCasa: lightweight affinity and object capabilities in Scala
https://dl.acm.org/doi/abs/10.1145/2983990.2984042

[2]: Deterministic Concurrency Using Lattices and the Object Capability Model
https://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A1268459&dswid=6775

[3]: Habanero-Java: the new adventures of old X10
https://dl.acm.org/doi/abs/10.1145/2093157.2093165

[5]: Semantics course at TUM
https://campus.tum.de/tumonline/ee/ui/ca2/app/desktop/#/slc.cm.reg/student/modules/detail/light/456513/study-year/1615

[6]: Tyranny of the Types: Curse or Blessing seminar at TUM
https://www.cs.cit.tum.de/pl/lehre/sommersemester-22/seminare/the-tyranny-of-types-curse-or-blessing/