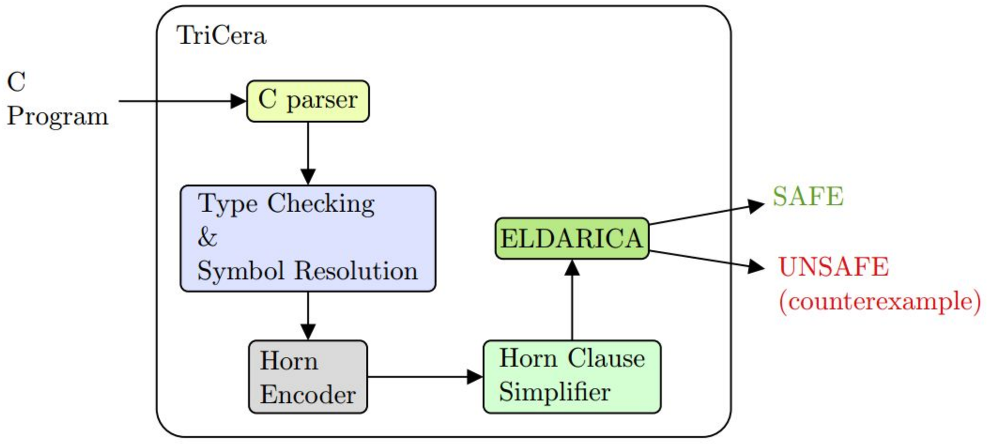
Extending TriCera to parse ACSL annotations

Background

Formal methods are becoming increasingly common in the industry and in the design and verification of embedded systems. Unlike testing, formal verification aims to prove the *absence* of bugs in a system. The widely used DO-178C standard of the avionics industry and the ISO26262 standard used in the automotive industry both recommend formal verification of safety critical systems to supplement testing.

One approach to formal verification is model checking. TriCera is a model checker for C programs, and it attempts the formal verification of a given C program with respect to some specifications. These specifications can be both implicit (e.g., memory and type safety) and explicit (through assertions and annotations).

TriCera takes a subset of C programs and encodes them in Horn clauses (see this thesis for details), which are then fed into a Horn solver such as Eldarica which checks if the generated clauses are solvable. If the clauses are concluded to be unsafe, a counterexample trace is provided as justification.



ACSL (ANSI/ISO C Specification Language) is used when specifying functional properties of C programs in the Frama-C framework. These are added as annotations to C programs in order to specify *pre/post-conditions* of functions, *loop invariants*, and *assertions* at specific program locations.

```
/*@ ensures \result >= x && \result >= y;
ensures \result == x || \result == y;
*/
int max (int x, int y) { return (x > y) ? x : y; }
```

(example is from https://frama-c.com/download/acsl-tutorial.pdf)

These annotations are only used during verification and are not part of the compiled program. See Frama-C and ACSL documentation for more details and tutorials.

Goals of the Thesis

The main goal is to extend TriCera so that it can parse (a subset of) ACSL annotations appearing in C source code.

The motivations for this work are

- to be able to directly verify ACSL annotated programs in TriCera with some advantages, such as not requiring the specification of loop invariants,
- to pave the way for future work such as automatic generation of (missing) annotations and loop invariants to be used by deductive verification tools.

The work will require you to get familiar with the C parser and Horn encoder of TriCera, and extend them with correct translations from ACSL annotations into Horn clauses. The performance of the implementation should be evaluated using benchmarks.

Relevant Background and Courses

Knowledge in compiler construction, formal methods, and logic will be required for this project. Those topics are partly covered in the following courses at Uppsala University; additional reading material will be provided to get a more complete background.

- 1DL321, Compiler Design I
- 1DL330, Functional Programming I
- 1DT034, Programming Theory
- 1DL481, Algorithms and Data Structures III
- 1DL500, Automata and Logic (currently not given)

TriCera is written in Scala, so knowledge of Scala and functional programming concepts would be helpful.

Practical Things

This project can be carried out by one student. The work will be carried out at the IT Department, or by the student working from home. Supervision will be provided through regular meetings with supervisors at the IT Department:

Contacts: Zafer Esen (zafer.esen@it.uu.se) and Philipp Rümmer (philipp.ruemmer@it.uu.se)

If you are interested in this project, please contact the above people; your application should include a CV.