Correct-by-construction Type Checking for Substructural Type Systems

Vince Szabó
V.Szabo-2@student.tudelft.nl
Supervisors: Jesper Cockx, Sára Juhošová

Introduction

Type systems provide static guarantees about programs.

Substructural type systems provide guarantees about the number of times each value is used.

Type checkers enforce the rules of the type system.

The guarantees of type systems can be proven.

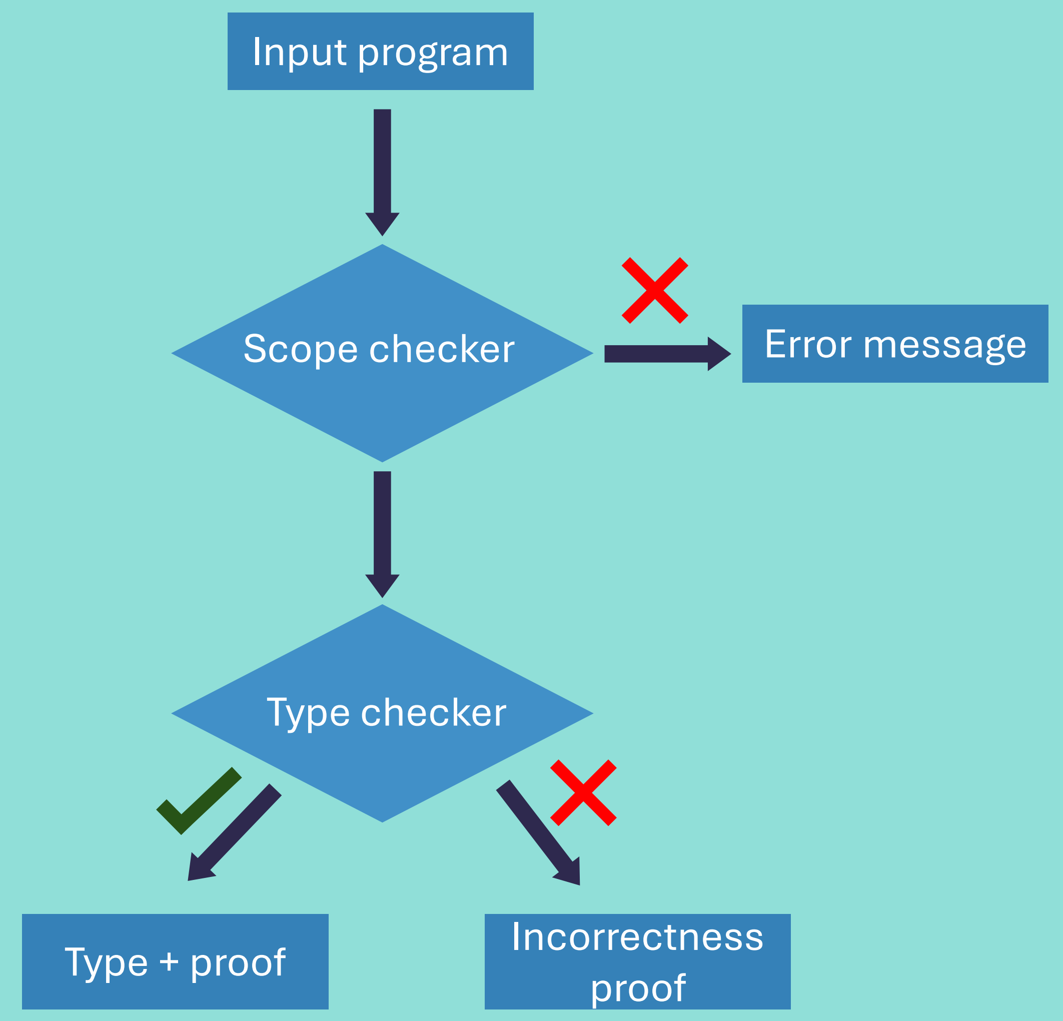
But how do we know that the type checker really follows the typing rules?

Method

Use Agda's type system to make it *impossible* to write an incorrect type checker.

Use a toy language derived from Walker's work [1], extended with affine and relevant types.

Structure



References
[1]: David Walker. 2004. Substructural Type Systems. In Advanced Topics in Types and Programming Languages, Benjamin C. Pierce (Ed.). The MIT Press, Chapter 1, 30–36.

Results

It works! (sound and complete)

~1300 lines of Agda, with runnable examples.

35s to type check the type checker.

Evaluation

Offers a very high degree of certainty.

Takes more time to develop than classical type checkers, harder to extend.

Proposed use cases: proof assistants, safety critical languages

Future work

- Complete scope checker
- Interpreter
- Prove guarantees of the type system