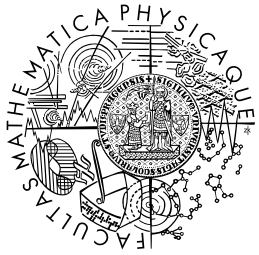

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Tomáš Husák

Improving Type Inference in the C# Language

Department of Distributed and Dependable Systems

Supervisor of the master thesis: Mgr. Pavel Ježek, Ph.D.
Study programme: Computer Science
Study branch: Software Systems

Prague 2024

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

TODO: Dedication.

Title: Improving Type Inference in the C# Language

Author: Tomáš Husák

Department: Department of Distributed and Dependable Systems

Supervisor: Mgr. Pavel Ježek, Ph.D., Department of Distributed and Dependable Systems

Abstract:

TODO: Abstract.

Keywords: Type Inference C# Roslyn

Contents

Introduction

Note: Describe what is type inference.

Statically typed languages have many advantages like revealing bugs in compilation time or performance. To achieve these benefits, the languages demand type annotations from a programmer. These type annotations define an actual type of variable during runtime protecting to make operations on incompatible data. Because code usually contains a lot of variables whose type has to be known during compilation time, type inference was introduced to eliminate type annotations that can be deduced from a context. Type inference tries to deduce a type of a variable using a context, where the variable is used. That's used operations and interactions with other parts of the code.

Note: Describe type inference in C#.

C# is a statically typed language whose type system, besides common primitives and classes known from other languages, contains generics. Generics are used for parametrizing types in order to create reusable code(e.g. containers). C# generics parametrizes types by other types. Main feature of the C# type inference is getting rid of type arguments in cases, where the arguments can deduces from a context. Despite of type inference is a very useful feature, possible scenarios where it can be applied is restrictive in comparison with other languages.

Note: Compare it with type inference in Rust or Haskell as an example of Hindel-Millner type inference.

As an example of advanced type inference we can mention Rust language. Although the type system has differences with C# type system, the type inference is done across multiple statements which is much more powerful than the former one. One of that reason regards specifics of type system, which enables to use Hindle-Millner type inference. Traditional Hindle-Milner type inference is defined in Hindle-Millner type system which has different characteristics from C# or Rust. Although the most powerful is in that type system, it can be adjusted to work in type systems in already mentioned languages.

Note: Describe C# specification

C# type inference is a variant of Hindle-Millner type inference and the algorithm can be found in the language specification. The specification consists of all language features described independently on the compiler. As the language evolves, the specification also changes. These changes are done publicly on a Github repository to offer participation of creating specification to wider community.

Note: Describe Roslyn.

Current implementation of C# type inference is contained in Roslyn. Roslyn is open source C# and VB compiler developed by Microsoft and community. Since it is open source, it is possible to investigate and participate on implementing new features to C# language.

Note: Mention CSharplang repo, community, and describe a process of accepting lang changes.

A common process to make a new possible language feature to be merged into Roslyn is making a proposal and a prototype. A proposal is a description of the new feature consisting of motivation, detailed description, needed language spec-

ification changes and other possible alternatives. The proposal is published in discussions where community can share their opinion. If the proposal is promising enough, the language committee will choose it for further discussions. The feature is added to the language if LDM accepts the proposal. After that, the implementation of that feature can be merged into Roslyn. During the proposal, a prototype is usually done to demonstrate the feature in wider examples.

Note: Goal of this thesis

Goal of this thesis is to create a proposal regarding to improve C# type inference in order to offer more power as we can see in other similar languages. To make the proposal more likely to be accepted by LDM committee, a prototype is created to estimate level of implementation difficulty in production C# compiler.

Note: Give an overview of chapters.

The first chapter describes Roslyn compiler together with theoretical background of type inference. The second chapter describes the scope of the language improvements based on community preferences. Then, It describes difficulties regarding architecture of C# language and the compiler implementation. At the end of this chapter, we describe goals of this work with given benefits. The third section consists of architecture design of implementation together with new type inference algorithm and changes made in the specification. The fourth chapter contains evaluation of the implemented feature. The last chapter discuss possible further features as a continuation and other possible interactions with already existing features.

1. Related work

TODO: Describe type inference in C#.

TODO: Describe Hindel-Millner type inference.

TODO: Describe type inference in Rust or Haskell (Mention related papers about type inference.

TODO: Describe Roslyn(Focused on the part where the type inference is done).

TODO: Mention related Github issues and csharplang repo.

2. Problem analysis

TODO: Describe outputs of this work(Proposal and prototype). Why these outputs are necessary.

TODO: Describe the set of related issues.

TODO: Describe the selection and scope of this work based on the issues and other factors.

TODO: Describe problems of C# lang architecture which prohibits some advanced aspects of type inference.

TODO: Describe goals of the work and explain benefits of proposed changes.

3. Solution

TODO: Describe process of making proposal and the prototype.

TODO: Describe partial method type inference.

TODO: Describe constructor type inference.

TODO: Describe generic adjusted algorithm for type inference.

TODO: Describe decisions of proposed change design.

TODO: Describe changed parts of C# standard.

4. Evaluation

TODO: Describe achieved type inference. Mention interesting capabilities.

TODO: Note about the performance.

TODO: Links to csharplang discussions.

5. Future improvements

TODO: Mention next steps which can be done.

TODO: Discuss which steps would not be the right way(used observed difficulties).

Conclusion

TODO: Describe issue selection.

TODO: Describe proposed changes in the lang.

TODO: Describe the prototype and proposal.

TODO: Mention csharplang discussions.

TODO: Mention observed future improvements.

Bibliography

List of Figures

List of Tables

List of Abbreviations

A. Attachments