Design and Development of a TypeChecker for RSL* Project Plan

Shreyas Ramanathan(s220281)
Co-Supervisor: Alceste Scalas
Co-Supervisor: Anne Elisabeth Haxthausen

05/02/2024 - 05/08/2024

1 Introduction

The detailed project plan outlines steps from research to final testing for developing a typechecker for RSL and RSL*. Initial phases focus on selecting appropriate programming languages and understanding RSL grammar. Design stages involve creating the lexer and parser for RSL, later extending these tools to accommodate RSL* specifics. Testing phases ensure these components work as intended. Advanced typechecking introduces complex type validations, followed by refinement.

2 Background

Type checking is a fundamental aspect of verifying software specifications, ensuring that the components of a software system interact correctly according to their defined types. However, the manual verification process is not only time-consuming but also prone to human error, necessitating the development of an automated typechecker. The significance of this research lies in its potential to streamline the software development process by automating the verification of type correctness in RSL and RSL* specifications. By addressing the ambiguities and complexities of these languages, the project aims to enhance their accessibility and reliability, contributing to the development of error-free software systems.

3 Tasks

The tasks below provide a skeleton of the flow of the thesis work over the course of its duration. It enlists the milestones that are required to be covered. These are as follows:

3.1 Study of RSL and RSL* (February)

This phase is dedicated to establishing a solid understanding of RSL (RAISE Specification Language) and its extension, RSL*. This focuses mainly on learning about specification language, especially RSL. Then moving on to learning about RSL* and key differences between them.

3.2 Learning F# and Scala (First week of March)

- **Objective:** Master the basics of F# and Scala to determine their suitability for implementing the typechecker.
- **Description:** Through tutorials, courses, and hands-on practice, learn the syntax, type systems, and functional programming paradigms of F# and Scala.
- **Expected Results:** Understanding and gaining significant knowledge on both languages to be able to develop Lexer, Parser and Typecheckers, especially for RSL/RSL*.

3.3 Identify and Work on Problematic Constructs (Second week of March)

- **Objective:** Highlight problematic and/or ambiguous constructs common to RSL and RSL*
- **Description:** Analyze the grammar and usage of constructs that are present in both languages, in order to generate ASTs wihtout much duplication or mutiple traversals.
- **Expected Results:** A list of problematic constructs with proposed resolutions or workarounds to ensure clear and unambiguous parsing and type checking.

3.4 Experimentation Handling Constructs in Both Languages(Second - Third Week of March)

- **Objective:** Experiment with implementations to best handle identified problematic constructs.
- **Description:** This phase involves a comparative analysis of F# and Scala. Developing small prototypes or code snippets in F# or Scala to experiment with various approaches for handling the problematic constructs. This will go hand-in-hand with the previous task.
- **Expected Results:** A refined approach for dealing with ambiguous constructs, informed by practical experimentation and testing. Also deciding which language to choose moving forward.

3.5 Build AST (Design) (Third week of March - First Week of April)

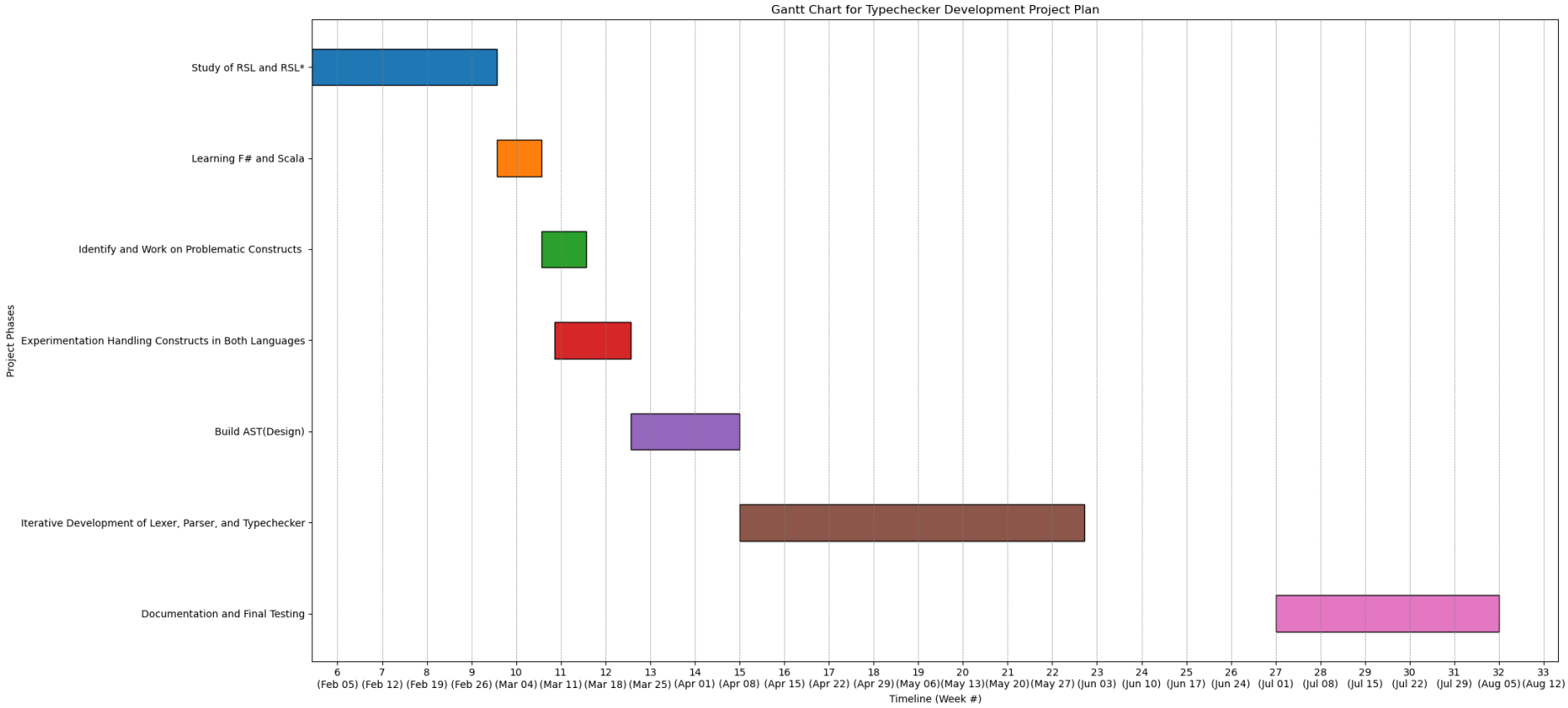
- **Objective:** Design the AST structure manually/programmatically to accurately represent RSL and RSL* syntax.
- **Description:** Outline the nodes and structure of the AST to encompass all necessary constructs from RSL and RSL*, focusing on modularity and extensibility.
- **Expected Results:** A comprehensive AST design that can effectively represent any construct from RSL and RSL* in a structured and navigable format.

3.6 Iterative Development of Lexer, Parser, and Typechecker (Second week of April - End of May)

- **Objective:** Develop the lexer, parser, and typechecker in phases, starting with simple constructs and then advanced constructs.
- **Description:** Begin with basic constructs and iteratively adding more complexity, refining the lexer, parser, and typechecker with each iteration. This tasks also contains systematic testing by creating test cases and analyse the code implementation.
- **Expected Results:** A series of progressively more capable versions of the lexer, parser, and typechecker, each iteration introducing new features and improvements. The end product will be a complete implementation targeting all intended constructs of RSL/RSL*.

3.7 Documentation and Final Testing (First week of July)

- **Objective:** Writing a thesis presenting the supported language and documenting the type checker and its development thoroughly.
- **Description:** The thesis will contain the entire process of the thesis workflow; starting from the initial phase of learning the language, mentioning about RSL and RSL*, choosing a language to build the typechecker in, until the complete implementation of the typechecker.
- **Expected Results:** An extensive thesis report.


Gantt Chart for Typechecker Development Project Plan
Project Phases
Study of RSL and RSL*
Learning F# and Scala
Identify and Work on Problematic Constructs
Experimentation Handling Constructs in Both Languages
Build AST(Design)
Iterative Development of Lexer, Parser, and Typechecker
Documentation and Final Testing
6 (Feb 05)
7 (Feb 12)
8 (Feb 19)
9 (Feb 26)
10 (Mar 04)
11 (Mar 11)
12 (Mar 18)
13 (Mar 25)
14 (Apr 01)
15 (Apr 08)
16 (Apr 15)
17 (Apr 22)
18 (Apr 29)
19 (May 06)
20 (May 13)
21 (May 20)
22 (May 27)
23 (Jun 03)
24 (Jun 10)
25 (Jun 17)
26 (Jun 24)
27 (Jul 01)
28 (Jul 08)
29 (Jul 15)
30 (Jul 22)
31 (Jul 29)
32 (Aug 05)
33 (Aug 12)
Timeline (Week #)