**PROJECT PLAN**

A graph with different colored squares

Description automatically generatedThe 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.

Figure 1: Gantt Chart for the Milestones of the Thesis

* Research and Tool Selection (March 1 - March 7, 2024)
  + Objective: Select the most suitable programming language and tools.
  + Tasks: Evaluate F# and Scala; assess libraries for parsing and type checking.
  + Outcome: Decision on programming language and toolkit.
  + Description: This phase involves a comparative analysis of F# and Scala, focusing on language features like functional programming support, type system complexity, and available libraries for syntax parsing. The goal is to identify the language that provides the optimal balance of functionality, ease of use, and community support for developing a typechecker.
* Understanding RSL and RSL\* Grammar (March 8 - March 15, 2024)
  + Objective: Thoroughly understand the grammar of RSL and RSL\*.
  + Tasks: Study RSL language documentation; differentiate between RSL and RSL\* syntax.
  + Outcome: Documentation summarizing key grammatical constructs of RSL and RSL\*.
  + Description: This milestone is dedicated to dissecting RSL's syntax and semantics, with an emphasis on identifying the subset relevant for RSL\*. It involves deep diving into official RSL documentation, academic papers, and existing implementations, aiming to construct a comprehensive overview that will guide the lexer and parser development.
* Initial Design (March 16 - March 21, 2024)
  + Objective: Design the architecture for the lexer, parser, and typechecker.
  + Tasks: Outline data flows and structures; define AST design.
  + Outcome: A set of design documents detailing system architecture and data models.
  + Description: The focus here is on creating a blueprint for the typechecker system, including the design of the AST and the flow of data through the lexer, parser, and typechecker components. This planning stage is crucial for ensuring that the system's architecture is robust, scalable, and capable of accurately analyzing RSL scripts.
* Implementing the Lexer for RSL (March 22 - April 14, 2024)
  + Objective: Develop a lexer to tokenize RSL syntax.
  + Tasks: Create regex patterns for tokens; code the lexer.
  + Outcome: A functional lexer capable of generating a stream of tokens from RSL scripts.
  + Description: Developing the lexer involves defining regular expressions to accurately tokenize the RSL language constructs. This component is critical for breaking down RSL scripts into manageable pieces that can be further analyzed by the parser.
* Developing the Parser for RSL (March 22 - April 14, 2024)
  + Objective: Build a parser that constructs an AST from tokens.
  + Tasks: Establish grammar rules; implement the parser.
  + Outcome: A parser that produces an AST from RSL tokens.
  + Description: This stage focuses on translating the stream of tokens generated by the lexer into an AST, using defined grammar rules that mirror RSL's syntax. The parser's ability to construct a correct and efficient AST is pivotal for the subsequent typechecking process.
* Extending Lexer/Parser for RSL\* (April 1 - April 23, 2024)
  + Objective: Adapt lexer and parser for RSL\* syntax.
  + Tasks: Update token patterns and grammar rules.
  + Outcome: Lexer and parser updated to support RSL\*.
  + Description: Building upon the lexer and parser developed for RSL, this phase extends both components to accommodate the additional constructs introduced by RSL\*. This involves refining the regex patterns and grammar rules to parse RSL\* syntax accurately.
* Testing Lexer and Parser (April 16 - April 26, 2024)
  + Objective: Validate the correctness of the lexer and parser.
  + Tasks: Develop test cases; identify and fix bugs.
  + Outcome: Comprehensive test report verifying lexer and parser functionality.
  + Description: Rigorous testing of the lexer and parser against a wide array of RSL and RSL\* scripts ensures that these components accurately tokenize and parse the language. This milestone is crucial for identifying and rectifying any defects, ensuring the reliability of the foundational tools for typechecking.
* Basic Typechecking (April 26 - May 31, 2024)
  + Objective: Implement basic type checking for RSL expressions.
  + Tasks: Define type rules; code basic type checking logic.
  + Outcome: A typechecker capable of validating basic RSL types.
  + Description: The development of the typechecker begins with implementing logic to verify the types of basic RSL expressions. This involves coding the rules that govern RSL's type system and applying these rules to the AST to ensure type correctness.
* Advanced Typechecking (May 11 - June 18, 2024)
  + Objective: Extend typechecking to complex RSL\* constructs.
  + Tasks: Incorporate advanced type rules; handle complex type scenarios.
  + Outcome: Enhanced typechecker supporting complex RSL\* features.
  + Description: Building on the basic typechecker, this phase focuses on integrating advanced type rules specific to RSL\*, enabling the typechecker to handle more complex type scenarios and constructs. This involves a deeper analysis of RSL\*'s type system and the development of sophisticated type checking algorithms.
* Refinement (June 11 - July 2, 2024)
  + Objective: Optimize and polish the typechecker.
  + Tasks: Code optimization; performance benchmarking; documentation.
  + Outcome: Optimized typechecker, and finalized documentation.
  + Description: The final refinement stage is dedicated to enhancing the typechecker's efficiency and usability. Subtasks include optimizing the code for performance, conducting benchmark tests to ensure the typechecker operates efficiently, and creating comprehensive documentation. This phase ensures the typechecker is not only accurate but also user-friendly and well-documented, facilitating future development and use.
* Documentation (July 3 - July 31, 2024)
  + Objective: Document the typechecker system thoroughly.
  + Tasks: Write Thesis report for the implementation.
* Final Testing and Wrap-up (July 10 - July 17, 2024)