Preface

The Art and Science of Parsing

Parsing is a fundamental concept in computer science that bridges the gap between human-readable text and machine-processable structures. At its core, parsing is the process of analyzing a sequence of symbols (like text) according to the rules of a formal grammar.

Whether you're building a programming language, a configuration file parser, or a domain-specific language for a specialized application, understanding parsing techniques is essential. This knowledge empowers you to create tools that can interpret and process structured text in meaningful ways.

Why Parsing Expression Grammars?

Parsing Expression Grammars (PEGs) represent a powerful approach to syntax analysis that combines the expressiveness of context-free grammars with the efficiency and predictability of recursive descent parsing.

Unlike traditional context-free grammars that can be ambiguous, PEGs are inherently unambiguous due to their ordered choice operator. This means that when multiple parsing rules could match, a PEG parser will always choose the first matching rule, eliminating ambiguity.

Key advantages of PEGs include:

- **Unambiguous parsing**: The ordered choice operator ensures a single valid parse for any input
- **Integrated lexical and syntactic analysis**: No separate lexer/tokenizer is needed
- **Powerful pattern matching**: Including lookahead and negative lookahead assertions
- **Intuitive notation**: Grammar rules that closely resemble EBNF notation
- **Recursive descent implementation**: Straightforward to implement and understand

About This Documentation

This documentation serves as both a reference guide and a tutorial for the TinyPEG library. We'll explore:

1. The theoretical foundations of Parsing Expression Grammars
2. The architecture and components of the TinyPEG library
3. Practical examples of building parsers for various use cases
4. A step-by-step guide to creating a complete tiny programming language

Current Status

The TinyPEG library is being actively developed. The current version includes:

- A fixed implementation of the core parsing components in `fixed_core.py` and `fixed_parsers.py`
- Working examples in the `examples` directory, including a standalone TinyCL implementation
- Comprehensive documentation and tutorials

For the most up-to-date information about the project's status, see the Project Status document.

Whether you're a student learning about parsing for the first time, a developer looking to build a custom parser, or an educator teaching language design, we hope this documentation provides valuable insights and practical guidance.

Let's embark on this journey into the fascinating world of parsing and language design!

"Programming languages are how programmers express and communicate ideas — and, occasionally, frustrations." — Simon Peyton Jones

---