Combinator Parsers

1. Basic

- **Parser**: A parser is a function that, given an input string, returns a set of possible parses. Each parse is typically a pair consisting of the parsed result and the remaining unparsed portion of the input.
- **Combinator**: A combinator is a higher-order function that takes one or more parsers as arguments and returns a new parser.

2. Formally defined parsers

Let P be a parser with the type signature:

$$P : \Sigma^* \to \mathcal{P}(\mathcal{R} \times \Sigma^*)$$

where Σ^* is the set of all possible input strings, $\mathcal{R}$ is the set of parse results, and $\mathcal{P}$ denotes the power set.

This means that a parser P takes an input string and returns a set of pairs. Each pair consists of a result and the remaining unparsed part of the input.

3. Combinators

- `Choice`: The choice combinator alt tries two parsers and returns the result of the first successful parser.

$$\text{alt}(P, Q) \ \textbf{def} \ \lambda s.\text{filter results from } (P(s) \cup Q(s)) \tag{1}$$

- `Sequence`: The sequence combinator seq combines two parsers such that the second parser is applied to the remaining input after the first parser succeeds.

$$\begin{aligned} \text{seq}(P, Q) \ \textbf{def} \ \lambda s.\{(r_1 \oplus r_2, s'') \mid (r_1, s') \in P(s) & \quad (2) \\ \text{and } (r_2, s'') \in Q(s')\} & \quad (3) \end{aligned}$$

  where $\oplus$ denotes some combination of the results r_1 and r_2.

- `Many`: The many combinator many applies a parser zero or more times.

$$\begin{aligned} \text{many}(P) \ \textbf{def} \ \lambda s.\{(\textit{results}, s') \mid & \quad (4) \\ \textit{results} \text{ is a list of results from zero or more applications of} & \quad (5) \\ P \text{ and } s' \text{ is the remaining input}\} & \quad (6) \end{aligned}$$

- `Many1`: The many1 combinator many1 applies a parser one or more times.

$$\text{many1}(P) \ \textbf{def} \ \text{seq}(P, \text{many}(P)) \tag{7}$$

- `Option`: The option combinator opt applies a parser and returns a default value if the parser fails.

$$\text{opt}(P, d) \ \textbf{def} \ \lambda s.\{(r, s) \mid (r, s) \in P(s)\} \cup \{(d, s) \mid \text{if } P(s) \text{ fails}\} \tag{8}$$

4. Formal definitions using Category Theory

In category theory, parsers can be modeled as functors between categories. For instance:

- **Category of Parsers**:
  - Objects: Parsers.
  - Morphisms: Combinators.
- **Functorial Composition**: A parser combinator can be viewed as a functor that maps parsers to new parsers. For example, the sequence combinator is a functor that maps a pair of parsers to a new parser.

Summary

The mathematical formalism of combinator parsers involves defining parsers as functions that map input strings to sets of parse results and remaining input. Combinators are higher-order functions that combine parsers in various ways. This formalism can be described using set theory, formal language theory, and category theory, providing a rigorous foundation for understanding and designing parser combinators.

Example: Parsing Addition

1. Category of Parsers

Let $\mathcal{C}$ be the category of parsers:

- **Objects**: Parsers P that parse integers.
- **Morphisms**: Combinators that combine parsers.

2. Functor definition

Define a functor F that maps a pair of integer parsers to a parser that parses their sum:

$$F : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$$

Given two parsers P_a and P_b that parse integers a and b, the functor F produces a new parser $F(P_a, P_b)$ that parses the sum $a + b$.

3. Action on objects

Let P_a and P_b be parsers:

$$F(P_a, P_b) = P_{a+b}$$

Where P_{a+b} is a parser that parses the sum $a + b$.

4. Action on morphisms

Let $\phi : P_a \to P_{a'}$ and $\psi : P_b \to P_{b'}$ be morphisms (combinators):

$$F(\phi, \psi) = \phi \circ \psi$$

Where $\circ$ denotes the composition of morphisms.

5. Example: Parsing Addition

Consider the parsers P_a and P_b that parse the integers $a = 3$ and $b = 4$. The functor F produces a new parser $F(P_a, P_b)$ that parses the sum:

$$F(P_a, P_b) = P_{3+4}$$

Thus, the parser P_{3+4} parses the integer 7.