I propose to add find patterns to pattern matching.

**Specification**

```ruby
find_pattern: Constant(*var, pat, ..., *var)
  | Constant[*var, pat, ..., *var]
  | [*var, pat, ..., *var]
```

This patterns are similar to array patterns, but it finds first match values from the given object.

```ruby
case ["a", 1, "b", "c", 2, "d", "e", "f", 3]
in [*pre, String => x, String => y, *post]
p pre #=> ["a", 1]
p x #=> "b"
p y #=> "c"
p post #=> [2, "d", "e", "f", 3]
end
```

Note that it doesn't support backtracking to avoid complexity. This means that the following code raises NoMatchingPatternError.

```ruby
case [0, 1, 2]
in [*, a, *] if a == 1
end
```

**Implementation**

- https://github.com/k-tsj/ruby/tree/find-pattern-prototype

**History**

#1 - 05/04/2020 09:03 AM - shyouhei (Shyouhei Urabe)

I am a noob in this area who don't understand what is good about it. Can you show us a bit more realistic example? The proposal kind of seems illustrative to me.

#2 - 05/04/2020 01:38 PM - shevegen (Robert A. Heiler)

Is this a bit like .scan(), but applied to the pattern matching style? (Not sure if I understood this completely but it reminds me a bit of using .scan() and MatchData on the result possibly.)