How about reintroducing `expr in pat`, as akr-san proposed in DevelopersMeeting20201026Japan.

The difference between `expr => pat` and new `expr in pat` is the return value of the expression.

```
# expr => pat
0 => a #=> void (succeeded)
0 => 1 #=> NoMatchingPatternError (failed)

# expr in pat
0 in a #=> true (succeeded)
0 in 1 #=> false (failed)
```

Motivation and use cases are described at [Feature #15865](#).

I pointed out that there is a concern that in this specification a user might overlook the pattern did not match, and changed the return value at [Feature #16355](#).

However, now we already have new "rightward assignment style" pattern matching syntax, so we can say that this problem is solved.

If `expr in pat` is accepted, I also propose that the return value of `expr => pat` on a successful match be the left-hand side value.
Because, in this case, it becomes more clear that the use of `expr => pat` is assignment.

Related issues:
Related to Ruby master - Feature #15865: `<expr> in <pattern>` expression Closed
Related to Ruby master - Feature #16355: Raise NoMatchingPatternError when `expr in pat` doesn't match Closed

Associated revisions
Revision 88f3ce12 - 12/13/2020 02:51 AM - ktsj (Kazuki Tsujimoto)
Reintroduce expr in pat [Feature #17371]

History

#1 - 12/06/2020 02:54 PM - ktsj (Kazuki Tsujimoto)
- Related to Feature #15865: `<expr> in <pattern>` expression added

#2 - 12/06/2020 02:55 PM - ktsj (Kazuki Tsujimoto)
- Related to Feature #16355: Raise NoMatchingPatternError when `expr in pat` doesn't match added

#3 - 12/10/2020 09:04 AM - mame (Yusuke Endoh)
- Target version set to 3.0

#4 - 12/13/2020 02:52 AM - ktsj (Kazuki Tsujimoto)
- Status changed from Open to Closed

Applied in changeset qii88f3ce12d32fbeb983b0950743c20253ea2d0c6.

Reintroduce expr in pat [Feature #17371]