Assigning local variables when using `case when regexp`

I want to use
```
case "str"
  when /s(?<mid>.)r/    
    p mid
end
```
instead of
```
case
  when /s(?<mid>.)r/ =~ "str"
    p mid
end
```
I also do not like using $1.

This feature is extremely useful when there are a lot of whens.

---

**History**

#1 - 04/17/2020 08:53 AM - nobu (Nobuyoshi Nakada)

It is because when calls === method but not =~.

#2 - 04/17/2020 10:20 AM - sawa (Tsuyoshi Sawada)

I have met such use cases, and agree this would be useful.

Alternatively, in order to access even unnamed matches, assigning the last matched data to a variable with the following syntax may be useful.

```
case "str"
  when /s(?<mid>.)r/ => match_data
    match_data[:mid] # => "t"
end
```
```
case "str"
  when /s().r/ => match_data
    match_data[1] # => "t"
end
```

#3 - 04/17/2020 11:11 AM - shyouhei (Shyouhei Urabe)

sawa (Tsuyoshi Sawada) wrote in #note-2:

Alternatively, in order to access even unnamed matches, assigning the last matched data to a variable with the following syntax may be useful.

```
case "str"
  when /s().r/ => match_data
    match_data[1] # => "t"
end
```

That's too confusing. The following code works today.

```
case "str"
in /s().r/ => match_data
  match_data[1] # => "t"
end
```
sawa wrote:

I have met such use cases, and agree this would be useful.

I don't disagree in the sense that there may be valid use cases, but the syntax is kind of weird:

/s(<mid>.)r/

I use regexes a lot of course, since they are useful. But do we really gain a lot from making what used to be simple, harder?

I understand that you may avoid an extra step (assignment, after the regex match), but to me personally I find that style so much harder to read, compared to:

```
if /foo.+/
  mid = $1.to_s.dup # or something like that
end
```

Admittedly I am very much a very oldschool-ruby person. ;)

By the way, while I personally do not really like $ variables, I actually use them a LOT, whereas I rarely use match_data[] syntax style. Dunno why, perhaps a habit, but the $1 $2 etc.. are one of the few (semi) global variables that I like. (I write "semi" because they tend to be more volatile, which is why I may tend to use .dup like a semi-crazy person a lot, rather than fix a regex or handle nils - I .to_s.dup all the things! ;)

---

Matz.

Implicit assignment from regular expression matches has been a bad idea. I shouldn't have merged it at the first moment. I'd like to remove it altogether if I have a chance. So I don't like to enhance it any further.

Matz.