There should be `Regexp#to_regexp`, just as there is `Array#to_ary` and `String#to_str`.

```ruby
p [].to_ary  #=> []
p '' .to_str  #=> ""
p // .to_regexp  # undefined method `to_regexp'...
```

The use case is code like this:

```ruby
if o.respond_to?(:to_ary)
  # do something with o.to_ary
elsif o.respond_to?(:to_str)
  # do something with o.to_str
elsif o.respond_to?(:to_regexp)
  # can't do this today
  # do something with o.to_regexp
```

The workaround is to use `Regexp.try_convert`. `Regexp.try_convert` accepts either a Regexp or an object that responds to `#to_regexp`; so this code works fine (and is in some ways better):

```ruby
elsif re = Regexp.try_convert(o)
  # do something with o
```

Still, that `Regexp` does not respond to `#to_regexp` surprised me. Does it surprise anyone else?

---

**History**

#1 - 05/16/2014 12:32 PM - wconrad (Wayne Conrad)

- File 0001-re-c-to_reg_to_regex-add-to_regexp.patch added

#2 - 12/17/2016 11:29 PM - justcolin (Colin Fulton)

I know this is an old issues, but this also surprised me.

#3 - 12/18/2016 02:34 AM - nobu (Nobuyoshi Nakada)

- Description updated

```ruby
case
  when ary = Array.try_convert(o)
    # do something with ary
  when str = String.try_convert(o)
    # do something with str
  when re = Regexp.try_convert(o)
    # do something with re
  else
    # do other thing
end
```

#4 - 02/22/2017 07:06 AM - matz (Yukihiro Matsumoto)

- Status changed from Open to Rejected

Is there any concrete use-case? Consistency is not the best reason.

This proposal leads against Duck typing.

Matz.