Ruby 1.8 - Bug #3550

Floating Point Representation Prevents Raising to Fractional Power

07/08/2010 04:32 PM - mspandit (Milind Pandit)

Status:	Open
Priority:	Normal
Assignee:	
Target version:	
ruby -v:	

Description

=begin
Not sure why this literal float can be raised successfully to a fractional power, but when assigned to a variable, it returns NaN.

$ ruby --version
ruby 1.8.7 (2009-06-12 patchlevel 174) [universal-darwin10.0]
$ irb
-2.21792114695341 ** 0.1
=> -1.08291560040828

```
-2.21792114695341 ** 1.1
=> -2.40182141051126
a = -2.21792114695341
=> -2.21792114695341
a ** 0.1
=> NaN # Expected: -1.08291560040828
a ** 1.1
=> NaN # Expected: -2.40182141051126
-2.21792114695341.to_f ** 1.1
=> NaN # Expected: -1.08291560040828
-2.21792114695341.to_f ** 0.1
=> NaN # Expected: -2.40182141051126
a - -2.21792114695341
=> 0.0
(a - -2.21792114695341).zero?
=> true
=end
```

Related issues:

Related to Ruby trunk - Bug #3746: Incorrect Float subtraction	**Rejected**	**08/26/2010**

History

#1 - 07/08/2010 07:41 PM - Eregon (Benoit Daloze)

=begin
On 8 July 2010 10:32, Milind Pandit redmine@ruby-lang.org wrote:

> Backport #3550: Floating Point Representation Prevents Raising to Fractional Power
> http://redmine.ruby-lang.org/issues/show/3550
>
> Author: Milind Pandit
> Status: Open, Priority: Normal
>
> Not sure why this literal float can be raised successfully to a fractional power, but when assigned to a variable, it returns NaN.
>
> $ ruby --version
> ruby 1.8.7 (2009-06-12 patchlevel 174) [universal-darwin10.0]
> $ irb
> -2.21792114695341 ** 0.1
> => -1.08291560040828
>
> ```
> -2.21792114695341 ** 1.1
> ```

```
=> -2.40182141051126
a = -2.21792114695341
=> -2.21792114695341
a ** 0.1
=> NaN # Expected: -1.08291560040828
a ** 1.1
=> NaN # Expected: -2.40182141051126
-2.21792114695341.to_f ** 1.1
=> NaN # Expected: -1.08291560040828
-2.21792114695341.to_f ** 0.1
=> NaN # Expected: -2.40182141051126
a - -2.21792114695341
=> 0.0
(a - -2.21792114695341).zero?
=> true
```

Because the answer is complex, and when you use both literal, the precedence of #** is higher than #-@

```
(-2.21792114695341) ** 1.1
=> NaN
But
-2.21792114695341 ** 1.1
=> -2.40182141051126
```

In 1.9, Complex are in the core and so:

```
(-2.21792114695341) ** 1.1
=> (-2.2842679034439537-0.742203633301588i)
```

I believe it is possible in 1.8, but "require 'complex'" did not resolve this for me.

=end