

Ruby - Bug #2756

Issues with Math and Complex behavior on 1.9

02/18/2010 04:10 AM - brixen (Brian Shirai)

Status:	Rejected	
Priority:	Normal	
Assignee:	mrkn (Kenta Murata)	
Target version:		
ruby -v:	ruby 1.9.2dev (2010-02-18 trunk 26704) [i386-darwin9.8.0]	Backport: 2.3: UNKNOWN, 2.4: UNKNOWN

Description

This ticket aggregates several issues with Math methods on 1.9. There are related tickets that either have not yet or do not, in my opinion, resolve these issues in a satisfactory manner. (see <http://redmine.ruby-lang.org/issues/show/1708>, and related to the behavior of Math <http://redmine.ruby-lang.org/issues/show/2189> and to 1.8 behavior <http://redmine.ruby-lang.org/issues/show/2754>)

1. There are behaviors that are inconsistent with 1.8

```
# On 1.8, the argument is coerced
$ ruby1.8.7 -v -e 'o = Object.new; def o.to_f; 0.5; end; p Math.atanh(o)'ruby 1.8.7 (2009-12-24 patchlevel 248) [i686-darwin9.8.0]
0.549306144334055
```

```
# On 1.9, the argument is not coerced
$ ruby1.9 -v -e 'o = Object.new; def o.to_f; 0.5; end; p Math.atanh(o)'
ruby 1.9.2dev (2010-02-18 trunk 26704) [i386-darwin9.8.0]
-e:1:in `atanh': can't convert Object into Float (TypeError)
    from -e:1:in `'
```

Q. Should 1.9 coerce arguments to Math methods?

```
# On 1.8, an ArgumentError is raised
$ ruby1.8.7 -v -e 'p Math.atanh("str")'
ruby 1.8.7 (2009-12-24 patchlevel 248) [i686-darwin9.8.0]
-e:1:in `atanh': invalid value for Float(): "str" (ArgumentError)
    from -e:1
```

```
# On 1.9, a TypeError is raised
$ ruby1.9 -v -e 'p Math.atanh("str")'
ruby 1.9.2dev (2010-02-18 trunk 26704) [i386-darwin9.8.0]
-e:1:in `atanh': can't convert String into Float (TypeError)
    from -e:1:in `'
```

Q. In this case, TypeError would appear more correct, so can the 1.8.7 behavior be changed? Also note that changing the 1.8.7 behavior would make it consistent with the behavior of atanh when requiring Complex (see <http://redmine.ruby-lang.org/issues/show/2754>)

2. There are behaviors that are inconsistent when requiring lib/complex.rb

```
# The original method raise a TypeError
$ ruby1.9 -v -e 'p Math.atanh(nil)'
ruby 1.9.2dev (2010-02-18 trunk 26704) [i386-darwin9.8.0]
-e:1:in `atanh': can't convert nil into Float (TypeError)
    from -e:1:in `'
```

```
# The new method attempts an undefined operation and consequently raises a NoMethodError
$ ruby1.9 -v -rcomplex -e 'p Math.atanh(nil)'
ruby 1.9.2dev (2010-02-18 trunk 26704) [i386-darwin9.8.0]
lib/complex.rb is deprecated
/Users/brian/devel/ruby19/install/lib/ruby/1.9.1/cmath.rb:196:in `atanh': undefined method `real?'
for nil:NilClass (NoMethodError)
    from -e:1:in `'
```

The same behavior is observed when passing a String.

Q. Should the behavior of `atanh` after requiring `lib/complex.rb` be the same for non-Complex inputs as before?

Also, requiring `lib/complex.rb` on 1.9 causes a warning: "`lib/complex.rb` is deprecated". But this is not entirely true. As best as I can understand from <http://redmine.ruby-lang.org/issues/show/1708>, it was never decided whether `complex.rb` should require `cmath.rb`. It appears that there are some behaviors acquired via `lib/complex.rb` that are not deprecated. In that case, this warning is confusing and misleading.

Q. Is `lib/complex.rb` deprecated or not? If it is, why is it deprecated and not removed? 1.9 already removes many libraries. Why is this one special and allowed to cause such confusion?

Q. Is there a definitive document that explains the policy and behavior of `Math` and `Complex` in 1.9?

To summarize the questions in this ticket?

- Q. Should 1.9 coerce arguments to `Math` methods?
- Q. Can we change the 1.8.7 behavior when raising exceptions to be both internally consistent and consistent with the behavior of 1.9 (Note that numerous changes to the exception raised have already been made in 1.8.5 -> 1.8.6 -> 1.8.7, so this request is not without precedent.) (see <http://redmine.ruby-lang.org/issues/show/2754>)
- Q. Should the behavior of `atanh` after requiring `lib/complex.rb` be the same for non-Complex inputs as before?
- Q. Is `lib/complex.rb` deprecated or not? If it is, *why* is it deprecated and not removed?
- Q. Is there a definitive document that explains the policy and behavior of `Math` and `Complex` in 1.9?

Thanks,
Brian

Related issues:

Related to Ruby - Bug #1708: require 'complex' Causes Unexpected Behaviour	Rejected	07/01/2009
Related to Ruby - Bug #2189: Math.atanh(1) & Math.atanh(-1) should not raise ...	Closed	10/10/2009
Has duplicate Ruby - Bug #3137: complex.rb changes exceptions of Math	Closed	04/12/2010

History

#1 - 02/18/2010 12:48 PM - mame (Yusuke Endoh)

=begin
Hi!

Q. Should 1.9 coerce arguments to `Math` methods?

I really want to agree, but it seems intended change for prohibiting string:

```
$ ruby18 -ve 'p Math.atanh("0.5")'  
ruby 1.8.8dev (2010-02-01 revision 26536) [i686-linux]  
0.549306144334055
```

```
$/ruby -ve 'p Math.atanh("0.5")'  
ruby 1.9.2dev (2010-02-18 trunk 26703) [i686-linux]  
-e:1:in `atanh': can't convert String into Float (TypeError) from -e:1:in '
```

The root cause is that explicit float conversion and implicit one are not distinguished. There are `to_i`, `to_int` and `to_f`, but no `to_flo`.

I think this can be fixed by adding `to_flo` in theory, but it may change behavior of many methods that take `Float` as arg. Needs much time to discuss, and the change is too big for 1.9 series.

Q. Can we change the 1.8.7 behavior when raising exceptions to be both internally consistent and consistent with the behavior of 1.9 (Note that numerous changes to the exception raised have already been made in 1.8.5 -> 1.8.6 -> 1.8.7, so this request is not without precedent.)

Shyouhei, please decide it.
IMO, TEENY version may accept such a small spec change, but patch-level might not.

Q. Should the behavior of `atanh` after requiring `lib/complex.rb` be the same for non-Complex inputs as before?

I agree in this case.
Ruby library generally tends to ignore such a type difference.
(see <http://redmine.ruby-lang.org/issues/show/2495>)
But cmath.rb attempts to replace the core Math module, so I think it has a responsibility to behave as similarly as possible. I'll work later.

Q. Is lib/complex.rb deprecated or not? If it is, *why* is it deprecated and not removed?

The main features of lib/complex.rb are embedded to core in 1.9, but some features (e.g., Numeric#im and complex-sensitive Math) are not. Complex-sensitive Math (just CMath, not replacing core Math) are moved to cmath.rb. Numeric#im and replacing core Math seemed to be deprecated. So lib/complex.rb is now just for compatibility.

This is just the fact. I don't know its reason.

Q. Is there a definitive document that explains the policy and behavior of Math and Complex in 1.9?

Of course, no it isn't, I guess ;-(

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#2 - 02/18/2010 03:23 PM - shyouhei (Shyouhei Urabe)

=begin
Hi,

Yusuke ENDOH wrote:

Q. Can we change the 1.8.7 behavior when raising exceptions to be both internally consistent and consistent with the behavior of 1.9 (Note that numerous changes to the exception raised have already been made in 1.8.5 -> 1.8.6 -> 1.8.7, so this request is not without precedent.)

Shyouhei, please decide it.
IMO, TEENY version may accept such a small spec change, but patch-level might not.

Agreed. I'd want to see this a spec change between versions.

Attachment: signature.asc
=end

#3 - 02/18/2010 03:31 PM - brixen (Brian Shirai)

=begin
Hi,

On Wed, Feb 17, 2010 at 10:23 PM, Urabe Shyouhei shyouhei@ruby-lang.org wrote:

Hi,

Yusuke ENDOH wrote:

Q. Can we change the 1.8.7 behavior when raising exceptions to be both internally consistent and consistent with the behavior of 1.9 (Note that numerous changes to the exception raised have already been made in 1.8.5 -> 1.8.6 -> 1.8.7, so this request is not without precedent.)

Shyouhei, please decide it.
IMO, TEENY version may accept such a small spec change, but patch-level might not.

Agreed. I'd want to see this a spec change between versions.

I would argue that the behavior of lib/complex.rb be considered a bug because the existing method is overridden but the existing method's behavior is changed incompatibly. Bugs can be fixed in patchlevels.

OTOH, while 1.8.8 is not well defined (at least I don't know where to look for a definition), it could be the target for this change. Of the two options, I'd really like to see it changed in 1.8.7.

Cheers,
Brian

=end

#4 - 04/13/2010 02:52 AM - mame (Yusuke Endoh)

- Target version changed from 1.9.2 to 2.0.0

=begin

=end

#5 - 06/26/2011 02:46 PM - naruse (Yui NARUSE)

- Status changed from Open to Assigned

- Assignee set to mame (Yusuke Endoh)

#6 - 07/05/2011 02:07 AM - mame (Yusuke Endoh)

- Assignee changed from mame (Yusuke Endoh) to mrkn (Kenta Murata)

Hello,

Q. Should 1.9 coerce arguments to Math methods?

Q. Can we change the 1.8.7 behavior when raising exceptions to be both internally consistent and consistent with the behavior of 1.9 (Note that numerous changes to the exception raised have already been made in 1.8.5 -> 1.8.6 -> 1.8.7, so this request is not without precedent.) (see <http://redmine.ruby-lang.org/issues/show/2754>)

Q. Should the behavior of atanh after requiring lib/complex.rb be the same for non-Complex inputs as before?

Q. Is lib/complex.rb deprecated or not? If it is, why is it deprecated and not removed?

Q. Is there a definitive document that explains the policy and behavior of Math and Complex in 1.9?

Though there is no official maintainer for both lib/complex.rb and lib/cmath.rb, I guess mrkn, keiju and tadf have a thorough knowledge of them. So pass this ticket to mrkn.

--

Yusuke Endoh mame@tsg.ne.jp

#7 - 02/17/2013 03:42 PM - mrkn (Kenta Murata)

- Target version changed from 2.0.0 to 2.6

#8 - 11/28/2017 09:22 AM - mrkn (Kenta Murata)

- Description updated

#9 - 12/25/2017 06:01 PM - naruse (Yui NARUSE)

- Target version deleted (2.6)

#10 - 03/15/2018 11:53 AM - mrkn (Kenta Murata)

- Status changed from Assigned to Rejected

I'm rejecting this issue because this is meaningless now.