Ruby master - Bug #12337

inconsistency between Fixnum#coerce and Bignum#coerce

05/01/2016 01:06 PM - akr (Akira Tanaka)

Status: Closed
Priority: Normal
Assignee: akr (Akira Tanaka)
Target version: ruby -v: ruby 2.4.0dev (2016-05-01 trunk 54866) [x86_64-linux]

Description
I found 1.coerce(2.0) is [2.0, 1.0] but (2**100).coerce(2.0) raises TypeError

This is a documented behavior.

But I think this is bad bahavior.

Fixnum and Bignum should work seamlessly.

For example, this exposes the platform is 32-bit or 64-bit.
(2**40) is Fixnum on 32-bit environment and Bignum on 64-bit environment.
So, (2**40).coerce(2.0) behaves differently: returns an array on 64-bit and raises TypeError on 32-bit platform.

I think the behavior of Bignum#coerce should be changed to match Fixnum#coerce (actually defined at Numeric).
#1 - 05/01/2016 02:22 PM - akr (Akira Tanaka)

I also found `1.coerce(2)` returns integers and `1.coerce(2**100)` returns floats.

```
% ./ruby -e 'p 1.coerce(2)'
[2, 1]
% ./ruby -e 'p 1.coerce(2**100)'
[1.2676506002282294e+30, 1.0]
% ./ruby -v
ruby 2.4.0dev (2016-05-01 trunk 54866) [x86_64-linux]
```

I think it should return integers because Fixnum and Bignum should work seamlessly.

#2 - 05/03/2016 09:10 AM - akr (Akira Tanaka)

- File `int-coerce.patch` added

I made a patch to fix this problem.

This implements `bignum.coerce(float)` to be `[float, float]` and `fixnum.coerce(bignum)` to be `[bignum, bignum]`.

```
% ./ruby -e 'fixnum = 3
bignum = 2**70
float = 5.0
[fixnum, bignum, float].each {|n|
  [fixnum, bignum, float].each {|n2|
    result = n1.coerce(n2).map {|n| n.class } rescue $!.class
    puts "#{n1.class}.coerce(#{n2.class}) => #{result.inspect}"
  }
} Fi
```

Note that a released version works as follows.

```
% ruby-2.3.1 -e 'fixnum = 3
bignum = 2**70
float = 5.0
[fixnum, bignum, float].each {|n|
  [fixnum, bignum, float].each {|n2|
    result = n1.coerce(n2).map {|n| n.class } rescue $!.class
    puts "#{n1.class}.coerce(#{n2.class}) => #{result.inspect}"
  }
} Fix
```

#3 - 05/03/2016 09:27 AM - Eregon (Benoit Daloze)

I had a question about `Bignum#coerce` in [http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/75176](http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/75176).

I think it has limited value to have `Bignum.coerce(Fixnum)` or `Fixnum.coerce(Bignum)` return `[Bignum, Bignum]` instead of just the operands, as it creates a Bignum in the Fixnum range, which is inconsistent with all other methods and might cause performance problems.

With [#12005](http://bludge.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/75176), this would be naturally the result with additionally no class distinction.
But indeed, Fixnum.coerce(Bignum) => [Float, Float] as currently is even worse probably (and shows how rarely {Bignum,Fixnum}#coerce are used in practice).

#4 - 05/03/2016 10:07 AM - akr (Akira Tanaka)

Bignum.coerce(Fixnum) is used to implement fixnum binop bignum.
(binop is binary operator such as +, -, etc.)

x binop y is implemented as follows if x's class doesn't know y's class.

u, v = y.coerce(x)
u binop v

This is the extension mechanism that we can implement new class
which can be used with numeric objects.
For example, 3 * Vector[4,5] works because Fixnum#* invokes
Vector#coerce which returns [Matrix::Scalar, Vector] and
invokes Matrix::Scalar#* which calculates actual scalar-vector multiplication.

Bignum#coerce is just an instance of this extension mechanism.
If bignum.coerce(fixnum) returns [fixnum, bignum] as you said,
"u binop v" can cause infinite recursion.

This mechanism is clearly useful for classes implemented in Ruby
script such as matrix.rb.
However matz also use it for builtin numeric classes: Fixnum, Bignum and Float.

#5 - 05/03/2016 11:36 AM - Eregon (Benoit Daloze)

Akira Tanaka wrote:

Bignum.coerce(Fixnum) is used to implement fixnum binop bignum.
(binop is binary operator such as +, -, etc.)

x binop y is implemented as follows if x's class doesn't know y's class.

[...]
However matz also use it for builtin numeric classes: Fixnum, Bignum and Float.

Thanks for the clarification.
In the specific case of Fixnum binop Bignum, I expect all Fixnum methods already know about Bignum and do not call coerce, isn't it?

I tried locally to raise an error when Fixnum.coerce(Bignum) is called and got only 1 failure in test-all
(due to rb_num_coerce_cmp called by ruby_num_interval_step_size(#size) on an Enumerator from Numerc#step with a Bignum step).

That said, I think your patch is an improvement until #12005 is merged.

#6 - 05/17/2016 06:26 AM - matz (Yukihiro Matsumoto)

- Assignee set to akr (Akira Tanaka)

bignum.coerce(float) raise an exception intentionally because it is difficult to determine which is bigger.
But it is only due to lack of effort, so it is OK to change (improve) the behavior.

fixnum.coerce(bignum) has similar situation.

Matz.

#7 - 05/17/2016 01:03 PM - akr (Akira Tanaka)

- Status changed from Open to Closed

This issue is closed because [Feature #12005] is merged.

matz said that backport for this issue to released branches is not appropriate.

#8 - 06/08/2016 05:16 AM - usa (Usaku NAKAMURA)

- Backport changed from 2.1: UNKNOWN, 2.2: UNKNOWN, 2.3: UNKNOWN to 2.1: DONTNEED, 2.2: DONTNEED, 2.3: DONTNEED

Files

| int-coerce.patch | 2.37 KB | 05/03/2016 | akr (Akira Tanaka) |