Ruby master - Bug #16518

Should we rationalize Rational's numerator automatically?

01/19/2020 07:02 AM - st0012 (Stan Lo)

Status:	Open		
Priority:	Normal		
Assignee:	mrkn (Kenta Murata)		
Target version:			
ruby -v:	2.6.5	**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

In this Rails issue we found that using BigDecimal as a Rational's numerator can have inconsistent behaviors when the denominator gets bigger. For example:

```
> 1 == BigDecimal(1)
=> true
> Rational(1, 1) == Rational(BigDecimal(1), 1)
=> true
> Rational(1, 6) == Rational(BigDecimal(1), 6)
=> true
> Rational(1, 60) == Rational(BigDecimal(1), 60)
=> false
```

So my question is, is this behavior expected?
If it's not expected, do we have a plan to fix it?
If it is, does it make sense to manually rationalize the numerator before passing it into the Rational call, in order to get a consistent result?

History

#1 - 01/20/2020 09:30 AM - Eregon (Benoit Daloze)

- Assignee set to mrkn (Kenta Murata)

This seems due to the fact Rational(BigDecimal, Integer) does not return a Rational but a BigDecimal!

That seems highly surprising to me, any Foo() "constructor" should return a Foo, isn't it?

Also a Float as numerator correctly produces a Rational, so the bug seems to be with BigDecimal as a numerator:

```
> Rational(BigDecimal(1), 2).class
=> BigDecimal
> Rational(BigDecimal(1).to_f, 2).class
=> Rational
```

#2 - 01/22/2020 08:42 AM - st0012 (Stan Lo)

Thanks for the quick response!

I also found that it returns BigDecimal if any of its arguments is a BigDecimal:

```
irb(main):005:0> Rational(BigDecimal(1), 1).class
=> BigDecimal
irb(main):006:0> Rational(BigDecimal(1), BigDecimal(1)).class
=> BigDecimal
irb(main):007:0> Rational(0, BigDecimal(1)).class
=> BigDecimal
irb(main):008:0>
```

#3 - 01/22/2020 09:27 AM - eict9620 (ZhengXian Qiu)

After a quick trace of the source code, the problem is BigDecimal is a subclass of Numeric but defined in Ruby.

The nurat_conver method check for argument type and convert to Rational, but the Numeric doesn't.
Ref: https://github.com/ruby/ruby/blob/5275d8bf4c43db9f057d24a26cf33ecd69f8b345/rational.c#L2632

When it calls the f_div method, it calculates it directly without converting them to Rational.

We can simply force convert the result, but I didn't think it is the best way:

```
return to_rational(f_div(a1, a2));
```