Ruby master - Bug #3434

Specs for coercion?

06/13/2010 03:29 PM - marcandre (Marc-Andre Lafortune)

Status:	Closed		
Priority:	Normal		
Assignee:	marcandre (Marc-Andre Lafortune)		
Target version:	2.6		
ruby -v:		**Backport:**	2.3: UNKNOWN, 2.4: UNKNOWN

Description

=begin
What are the official specs of coercion for mathematical classes?

I will take Matrix as an example, but my questions are meant to be for the general case.

My understanding is that for a obj.coerce(obj_2) should return [compatible_2, compatible] such that compatible2.send(some_operation, compatible) returns (if possible) a meaningful result.

Can we assume anything more about coercion? I'm asking because I find in test_matrix (@m1 being a Matrix):

```
def test_scalar_mul
s1 = @m1.coerce(1).first
assert_equal(Matrix1, (s1 * 1) * Matrix1)
assert_equal(Vector[2], s1 * Vector[2])
assert_equal(Matrix2, s1 * Matrix2)
o = Object.new
def o.coerce(x)
[1, 1]
end
assert_equal(1, s1 * o)
end
```

1) Should the first and last assert work? Are implementers of other mathematical classes mandated/encouraged to provide that level of compatibility? Are users of coerce encouraged (or guaranteed success) when doing any other operation than compatible2 * compatible or similar?

My feeling is that the only thing one should do with the results of coerce is to call an operation on the first element and pass the second element. Doing operations using only one of the two returned elements with another new object should yield undetermined results.

If this is the case, I feel the Matrix library would be best to assume that Scalar#* is called with a Matrix or Vector (remember that the Scalar is not meant to be used directly by anybody else than the Matrix lib), and these two assert would not work.

2) The second assert is clearly implementation dependant. In the current implementation, both Vector and Matrix use Matrix::Scalar as a temporary class, but that could change.

3) I am assuming that compatible doesn't have to be equal?, eql? or == to obj nor of the same class, and the same is true for compatible_2 vs obj_2, right? Thus the third assert isn't a spec guaranteed for all implementations.

4) Finally, a somewhat trivial question: should obj.coerce(obj_2) succeed if obj and obj_2 are of the same class (and presumably return [obj_2, obj])? Clearly this is not very useful, but the specs should be made precise.

This is the case for Numeric and is explicit in the documentation but fails for Matrix:

```
Matrix.I(2).coerce(Matrix.I(2))  # => TypeError: Matrix can't be coerced into Matrix
```

Coercion should always work for the trivial case of two objects of the same class, right?

Thanks.
--
Marc-André
=end

History

#1 - 07/17/2010 03:32 PM - yugui (Yuki Sonoda)

- Target version changed from 1.9.2 to 2.0.0

=begin

=end

#2 - 09/14/2010 03:58 PM - shyouhei (Shyouhei Urabe)

- Status changed from Open to Assigned

=begin

=end

#3 - 02/18/2013 04:19 AM - trans (Thomas Sawyer)

=begin
Regarding the coercion spec, I had need of coercion for Array the other day while implementing a set logic system. So I wondered if coercion can't be more general and not just isolated to Numerics. Is there a necessary reason coercion can't work in general for ((*operators*)) across ((*all classes*))?
=end

#4 - 02/24/2013 09:18 PM - ko1 (Koichi Sasada)

- Description updated

I'm not sure about this ticket.
Who can discuss about it?

#5 - 02/24/2013 09:18 PM - ko1 (Koichi Sasada)

- Category set to core

- Target version changed from 2.0.0 to 2.6

#6 - 04/18/2013 05:59 AM - naruse (Yui NARUSE)

- Tracker changed from Misc to Bug

#7 - 11/27/2017 11:48 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Feedback

All of this are just my opinion, but obviously, coercion is a "best effort" feature. I believe there is no conclusive spec for coercion even in matz's heart. Until we encounter a concrete case that doesn't work well, we can't think what is a desirable behavior.

> My feeling is that the only thing one should do with the results of coerce is to call an operation on the first element and pass the second element.

I agree with this. IMO, all but this (or sometimes, all including this) are not guaranteed.

If you have any concrete case that does not work well, it would be good to disclose it. (Marc-Andre, if your case is related to matrix.rb, you are now the maintainer, so you may fix it yourself :-)

#8 - 11/28/2017 03:45 AM - marcandre (Marc-Andre Lafortune)

- Status changed from Feedback to Closed

Thanks for your reply.
I agree that we need to base on concrete examples, so I'll close this.