The way <=> works on pretty much everything in Ruby is that if a <=> b return 0, 1, or -1, it completely determines the entire set of comparisons a==b, a>b, a<b, aa, b==a, b>a, b<a. (and if it doesn't, a==b/b==a will be both true or both false, everything else will raise exception or return false/nil)

Float Infinity in 1.9 but not 1.8 seems to violate that. Comparing it with strange things returns 1 if it's on the left, but raises exception in every other way.

inf = 1.0/0.0
inf <=> "foo" # => 1
"foo" <=> inf # ArgumentError: comparison of String with Float failed

This interacts even more strangely with very large bignums and the "if bignum converts to float, it equals that float" thing Ruby currently does [ruby-core:31376].

inf=1.0/0.0
huge=10**500

Consistent either way:
inf >= huge #=> true
huge <= inf #=> true
inf < huge #=> false
huge > inf #=> false

Consistent only with mathematical interpretation
(or with "equal if converts, except for special cases for infinities"):
inf <=> huge #=> 1
huge<=inf #=> -1
huge < inf #=> true
huge > inf #=> false

Consistent only with "equal if converts":
inf == huge #=> true
huge == inf #=> true
inf > huge #=> false
inf <= huge #=> true

Now I'd definitely prefer mathematical interpretation of floats, to "equal if converts", but this just doesn't make any sense no matter which way I look at it.
I completely agree that Math::Float <=> "foo" should return nil.

The current behavior is due to r23742 which wanted to address the fact that Float::Infinity <=> BigDecimal("1.0E500") was returning 0 (I think, see rubydev:38681)

To fix Float::Infinity <=> "foo", the minimum that must be done is:

diff --git a/numeric.c b/numeric.c
index eb3d4be..daa5d6d 100644
--- a/numeric.c
+++ b/numeric.c
@@ -1038,7 +1038,7 @@ flo_cmp(VALUE x, VALUE y)
break;

-default:
+default:
  if (isinf(a) && (!rb_respond_to(y, rb_intern("infinite?"))) ||
  !RTEST(rb_funcall(y, rb_intern("infinite?"), 0, 0)))) { if (a > 0.0) return INT2FIX(1); return INT2FIX(-1);

The fact that <=> is not consistent with <, etc, is also a problem that need to be fixed. Either the special treatment should be extended to the other comparison operators, or the special treatment for infinity should be removed from <=>

I believe the special treatment should be removed altogether:

diff --git a/numeric.c b/numeric.c
index eb3d4be..a6c5360 100644
--- a/numeric.c
+++ b/numeric.c
@@ -1038,11 +1038,6 @@ flo_cmp(VALUE x, VALUE y)
break;

-default:
+default:
  if (isinf(a) && (!rb_respond_to(y, rb_intern("infinite?"))) ||
  !RTEST(rb_funcall(y, rb_intern("infinite?"), 0, 0))) { if (a > 0.0) return INT2FIX(1);
  return INT2FIX(-1);
  } return rb_num_coerce_cmp(x, y, rb_intern("<=>")); return rb_dbl_cmp(a, b);

I understand the intent, but the fact is that Float::INFINITY is a very big value, but since it is the float representation of a lot of big real numbers, like 10^400, 10^40000 or even Infinity itself, I feel that r23742 introduces many inconsistencies. For example, currently:

1.0e200 ** 2 <=> BigDecimal("1.0e99999") # => 1

#2 - 07/25/2010 05:47 AM - nobu (Nobuyoshi Nakada)
- Status changed from Open to Closed
- % Done changed from 0 to 100

=begin
This issue was solved with changeset r28751.
Tomasz, thank you for reporting this issue.
Your contribution to Ruby is greatly appreciated.
May Ruby be with you.
=end

#3 - 07/25/2010 06:01 AM - nobu (Nobuyoshi Nakada)
- Status changed from Closed to Assigned
- Assignee set to yugui (Yuki Sonoda)

=begin
=end
#4 - 07/30/2010 01:50 AM - mame (Yusuke Endoh)
- Status changed from Assigned to Closed

=begin
Backported at r28788.
=end

#5 - 07/30/2010 06:54 AM - marcandre (Marc-Andre Lafortune)
- Status changed from Closed to Open
- Assignee deleted (yugui (Yuki Sonoda))

=begin
The patch fixes comparison with non numerics, but doesn't address the rest of the issues:

- inconsistency with mathematics
- inconsistency with other operators like <, <=, ..

Is there objection to removing the special test for infinity?

diff --git a/numeric.c b/numeric.c
index 740ef54..ed159ce 100644
--- a/numeric.c
+++ b/numeric.c
@@ -1039,15 +1039,6 @@ flo_cmp(VALUE x, VALUE y)
 break;
 default:
 - if (isinf(a) && (i = rb_check_funcall(y, rb_intern("infinite?"), 0, 0)) != Qundef) {}
 - if (RTEST(i)) {
 - int j = rb_cmpint(i, x, y);
 - if (a > 0.0) ? (j > 0 ? 0 : +1) : (j < 0 ? 0 : -1);
 - return INT2FIX(j);
 - } if (a > 0.0) return INT2FIX(1);
 - return INT2FIX(-1);
 - return rb_num_coerce_cmp(x, y, rb_intern("<=>")); } return rb_dbl_cmp(a, b);

=end

#6 - 07/30/2010 12:47 PM - mame (Yusuke Endoh)
- Target version set to 2.0.0

=begin
Hi,

The patch fixes comparison with non numerics, but doesn't address the rest of the issues:

Indeed. I thought nobu aimed to fix only the obvious wrong condition.

Is there objection to removing the special test for infinity?

It looks like a design issue rather than code bug. So I change this to 1.9.x.

I have no objection against removal of the code in trunk. Though, I like rather extend the special test to other operators than remove.

--
Yusuke Endoh mame@tsg.ne.jp
=end

#7 - 08/27/2010 01:59 PM - shyouhei (Shyouhei Urabe)
- Status changed from Open to Closed

=begin