Ruby master - Bug #17037

rounding of Rational#to_f

07/20/2020 05:41 PM - akr (Akira Tanaka)

Status:	Open		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:		**Backport:**	2.5: UNKNOWN, 2.6: UNKNOWN, 2.7: UNKNOWN

Description

I found a doubtful rounding behavior of Rational#to_f.

```
% ./ruby -ve '
a = 1r
e = Float::EPSILON.to_r
puts "e=#{e}"
n = 100
(0..n).each {|i|
  r = a + e * i / n
  f = r.to_f
  p [i, f, r]
}
'
ruby 2.8.0dev (2020-07-20T06:39:31Z master 935d0b3d05) [x86_64-linux]
e=1/4503599627370496
[0, 1.0, (1/1)]
[1, 1.0, (450359962737049601/450359962737049600)]
[2, 1.0, (225179981368524801/225179981368524800)]
[3, 1.0, (450359962737049603/450359962737049600)]
[4, 1.0, (112589990684262401/112589990684262400)]
[5, 1.0, (90071992547409921/90071992547409920)]
[6, 1.0, (225179981368524803/225179981368524800)]
[7, 1.0, (450359962737049607/450359962737049600)]
[8, 1.0, (56294995342131201/56294995342131200)]
[9, 1.0, (450359962737049609/450359962737049600)]
[10, 1.0, (45035996273704961/45035996273704960)]
[11, 1.0, (450359962737049611/450359962737049600)]
[12, 1.0, (112589990684262403/112589990684262400)]
[13, 1.0, (450359962737049613/450359962737049600)]
[14, 1.0, (225179981368524807/225179981368524800)]
[15, 1.0, (90071992547409923/90071992547409920)]
[16, 1.0, (28147497671065601/28147497671065600)]
[17, 1.0, (450359962737049617/450359962737049600)]
[18, 1.0, (225179981368524809/225179981368524800)]
[19, 1.0, (450359962737049619/450359962737049600)]
[20, 1.0, (22517998136852481/22517998136852480)]
[21, 1.0, (450359962737049621/450359962737049600)]
[22, 1.0, (225179981368524811/225179981368524800)]
[23, 1.0, (450359962737049623/450359962737049600)]
[24, 1.0, (56294995342131203/56294995342131200)]
[25, 1.0, (18014398509481985/18014398509481984)]
[26, 1.0, (225179981368524813/225179981368524800)]
[27, 1.0, (450359962737049627/450359962737049600)]
[28, 1.0, (112589990684262407/112589990684262400)]
[29, 1.0, (450359962737049629/450359962737049600)]
[30, 1.0, (45035996273704963/45035996273704960)]
[31, 1.0, (450359962737049631/450359962737049600)]
[32, 1.0, (14073748835532801/14073748835532800)]
[33, 1.0000000000000002, (450359962737049633/450359962737049600)]
[34, 1.0000000000000002, (225179981368524817/225179981368524800)]
[35, 1.0, (90071992547409927/90071992547409920)]
```

```
[36, 1.0000000000000002, (112589990684262409/112589990684262400)]
[37, 1.0000000000000002, (450359962737049637/450359962737049600)]
[38, 1.0000000000000002, (225179981368524819/225179981368524800)]
[39, 1.0000000000000002, (450359962737049639/450359962737049600)]
[40, 1.0, (11258999068426241/11258999068426240)]
[41, 1.0000000000000002, (450359962737049641/450359962737049600)]
[42, 1.0000000000000002, (225179981368524821/225179981368524800)]
[43, 1.0000000000000002, (450359962737049643/450359962737049600)]
[44, 1.0000000000000002, (112589990684262411/112589990684262400)]
[45, 1.0000000000000002, (90071992547409929/90071992547409920)]
[46, 1.0000000000000002, (225179981368524823/225179981368524800)]
[47, 1.0000000000000002, (450359962737049647/450359962737049600)]
[48, 1.0000000000000002, (28147497671065603/28147497671065600)]
[49, 1.0000000000000002, (450359962737049649/450359962737049600)]
[50, 1.0, (9007199254740993/9007199254740992)]
[51, 1.0000000000000002, (450359962737049651/450359962737049600)]
[52, 1.0000000000000002, (112589990684262413/112589990684262400)]
[53, 1.0000000000000002, (450359962737049653/450359962737049600)]
[54, 1.0000000000000002, (225179981368524827/225179981368524800)]
[55, 1.0000000000000002, (90071992547409931/90071992547409920)]
[56, 1.0000000000000002, (56294995342131207/56294995342131200)]
[57, 1.0000000000000002, (450359962737049657/450359962737049600)]
[58, 1.0000000000000002, (225179981368524829/225179981368524800)]
[59, 1.0000000000000002, (450359962737049659/450359962737049600)]
[60, 1.0000000000000002, (22517998136852483/22517998136852480)]
[61, 1.0000000000000002, (450359962737049661/450359962737049600)]
[62, 1.0000000000000002, (225179981368524831/225179981368524800)]
[63, 1.0000000000000002, (450359962737049663/450359962737049600)]
[64, 1.0000000000000002, (7036874417766401/7036874417766400)]
[65, 1.0000000000000002, (90071992547409933/90071992547409920)]
[66, 1.0000000000000002, (225179981368524833/225179981368524800)]
[67, 1.0000000000000002, (450359962737049667/450359962737049600)]
[68, 1.0000000000000002, (112589990684262417/112589990684262400)]
[69, 1.0000000000000002, (450359962737049669/450359962737049600)]
[70, 1.0000000000000002, (45035996273704967/45035996273704960)]
[71, 1.0000000000000002, (450359962737049671/450359962737049600)]
[72, 1.0000000000000002, (56294995342131209/56294995342131200)]
[73, 1.0000000000000002, (450359962737049673/450359962737049600)]
[74, 1.0000000000000002, (225179981368524837/225179981368524800)]
[75, 1.0000000000000002, (18014398509481987/18014398509481984)]
[76, 1.0000000000000002, (112589990684262419/112589990684262400)]
[77, 1.0000000000000002, (450359962737049677/450359962737049600)]
[78, 1.0000000000000002, (225179981368524839/225179981368524800)]
[79, 1.0000000000000002, (450359962737049679/450359962737049600)]
[80, 1.0000000000000002, (5629499534213121/5629499534213120)]
[81, 1.0000000000000002, (450359962737049681/450359962737049600)]
[82, 1.0000000000000002, (225179981368524841/225179981368524800)]
[83, 1.0000000000000002, (450359962737049683/450359962737049600)]
[84, 1.0000000000000002, (112589990684262421/112589990684262400)]
[85, 1.0000000000000002, (90071992547409937/90071992547409920)]
[86, 1.0000000000000002, (225179981368524843/225179981368524800)]
[87, 1.0000000000000002, (450359962737049687/450359962737049600)]
[88, 1.0000000000000002, (56294995342131211/56294995342131200)]
[89, 1.0000000000000002, (450359962737049689/450359962737049600)]
[90, 1.0000000000000002, (45035996273704969/45035996273704960)]
[91, 1.0000000000000002, (450359962737049691/450359962737049600)]
[92, 1.0000000000000002, (112589990684262423/112589990684262400)]
[93, 1.0000000000000002, (450359962737049693/450359962737049600)]
[94, 1.0000000000000002, (225179981368524847/225179981368524800)]
[95, 1.0000000000000002, (90071992547409939/90071992547409920)]
[96, 1.0000000000000002, (14073748835532803/14073748835532800)]
[97, 1.0000000000000002, (450359962737049697/450359962737049600)]
[98, 1.0000000000000002, (225179981368524849/225179981368524800)]
[99, 1.0000000000000002, (450359962737049699/450359962737049600)]
[100, 1.0000000000000002, (4503599627370497/4503599627370496)]
```

This sample program tries to convert rationals between

1.0 and 1.0 + Float::EPSILON to float.
Since there is no representable float values between them
(except 1.0 and 1.0 + Float::EPSILON),
I expect that
r.to_f returns 1.0 for r < X and
1.0 + Float::EPSILON for r > X
for some X.

But my expectation is not valid:

```
[34, 1.0000000000000002, (225179981368524817/225179981368524800)]
[35, 1.0, (90071992547409927/90071992547409920)]
```

I.e.

1 + Float::EPSILON.to_r * 34 / 100 <
1 + Float::EPSILON.to_r * 35 / 100

but

(1 + Float::EPSILON.to_r * 34 / 100).to_f >
(1 + Float::EPSILON.to_r * 35 / 100).to_f

I guess Rational#to_f should round the rational value to a nearest representable float value.

History

#1 - 08/26/2020 03:54 AM - akr (Akira Tanaka)

Gauche scheme interpreter has same issue and Kawai-san (the author of Gauche) investigate it.

http://blog.practical-scheme.net/gauche/20200722-ratnum-flonum