2015 年度 Ruby 開発プロジェクト助成金 最終報告

2016 年 3 月 4 日 斎藤ただし

概要

プロジェクトの作業はほぼ問題なく完了した。既存の Rational メソッドほぼすべてについて最適化を施し、数パーセント～3.6 倍以上のパフォーマンスの向上がマイクロベンチマークで確認できた。深刻なデグレーションもなく、Rational の全面的な高速化が達成された。

背景

古くからの有理数処理のためのライブラリである Rational は、Ruby 1.9.1 のリリースからその実装が C 言語に変更された。しかしその実装には未だに Ruby レベルのメソッド呼び出し(rb_funcall())が多く残されている。

これには、元々 Ruby による実装であった Rational を、C 言語によって再実装するにあたってスムーズであったり、高い互換性を達成できる、という利点はあったと考えられるものの、C 言語で全面的に書き換えられた今日、そのパフォーマンス上のデメリットは小さくない。

他方、Ruby1.9.2 からは Time の内部表現として採用され、2.1 からはリテラル表記が導入されるなど、Rational の重要性は相対的に高まり続けている。

本プロジェクトでは、Rational 実装における不要な rb_funcall()呼び出しを極力削り、より直接的な C による実装に置き換えることによって、Rational における全面的な最適化を図った。

実装

まず、四則演算である `Rational#+`、 `Rational#-`、 `Rational#*`、そして `Rational#/` の 4 つのメソッドと、それらが依存する処理について最適化を試みた。いずれも `rb_funcall()` が間接・直接に呼び出されていたので、それらを C ネイティブな処理に置き換えた。完全な変更点は後述の資料 3 に添付したが、その中でも典型的な作業例 `Ratinoal * Float` は以下のようなものであった。

```
diff --git a/rational.c b/rational.c
index 073b9da..30a1240 100644
--- a/rational.c
+++ b/rational.c
@@ -886,7 +886,7 @@ nurat_mul(VALUE self, VALUE other)
         }
     }
     else if (RB_TYPE_P(other, T_FLOAT)) {
-        return f_mul(f_to_f(self), other);
+        return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) * RFLOAT_VALUE(other));
     }
```

```
    else if (RB_TYPE_P(other, T_RATIONAL)) {
       {
```

ここでは `f_to_f()` が (`self` に対して) Ruby メソッドの `to_f` を、`f_mul()` が同じく Ruby メソッドの `*` をそれぞれ呼び出していた。しかしここでは Ruby が提供する多態性などは不要であるので、レシーバーを Rational オブジェクト決め打ちにした `nurat_to_f()` で前者を、C レベル演算子の `*` で後者を、それぞれ置き換えた。

さらに作業続けた結果、具体的には以下のようなケースで同様の最適化が可能であった。

- `Rational#+`
  - Rational + Float
  - Rational + Integer
  - Rational + Rational およびそれが依存する Integer#gcd
- `Rational#-`
  - Rational - Float
  - Rational - Integer
  - Rational - Rational
- `Rational#*`
  - Rational * Float
  - Rational * Bignum および Rational * 精度の高い(内部表現として Bignum を持つ)Rational
- `Rational#/`
  - Rational / Float
  - Rational / Bignum および Rational / 精度の高い(内部表現として Bignum を持つ)Rational
- `Rational#fdiv`
  - 引数が 0、1、それ以上、のそれぞれのケース
- `Rational#**`
  - Rational ** Fixnum
  - Rational ** Bignum
  - Rational ** Float
- `Rational#<=>`
  - Rational Rational
  - Rational Float
  - Rational Fixnum
- `Rational()`
  - Rational(Integer, Integer)
  - Rational(Bignum)
  - Rational(String)
    - 整数文字列
    - 小数文字列
    - 指数表現の小数文字列
- `Rational#==`

    - Rational == Rational
    - Rational == Bignum
- `Rational#coerce(Float)`
- `Rational#floor`
    - 引数なし
    - 引数あり
- `Rational#ceil`
    - 引数なし
    - 引数あり
- `Rational#round`
    - 引数なし
    - 引数あり
- `Rational#truncate`
    - 引数なし
    - 引数あり
- `Numeric#numerator`
- `Numeric#denominator`
- `Numeric#quo(Integer)`
- `Float#numerator`
- `Float#denominator`
- `Float#to_r`
    - 精度の低いケース
    - 精度の高いケース
- `Float#rationalize`
    - 正の数
    - 負の数
    - ゼロ
    - 正の数で引数あり
    - 負の数で引数あり
    - 負の数で精度が必要なケース
- `Rational#negaive?`
- `Integer#lcm`
- `Rational#-@`

評価

複数の観点で改善の評価を行った。なお評価に用いたマシンのスペック・環境等は以下である。

- ThinkPad X230
    - Intel Core i5-3210M CPU @ 2.50GHz
    - RAM: 16GB
- Debian GNU/Linux stretch (testing)
    - 全 CPU コアのポリシーを `performance` に設定

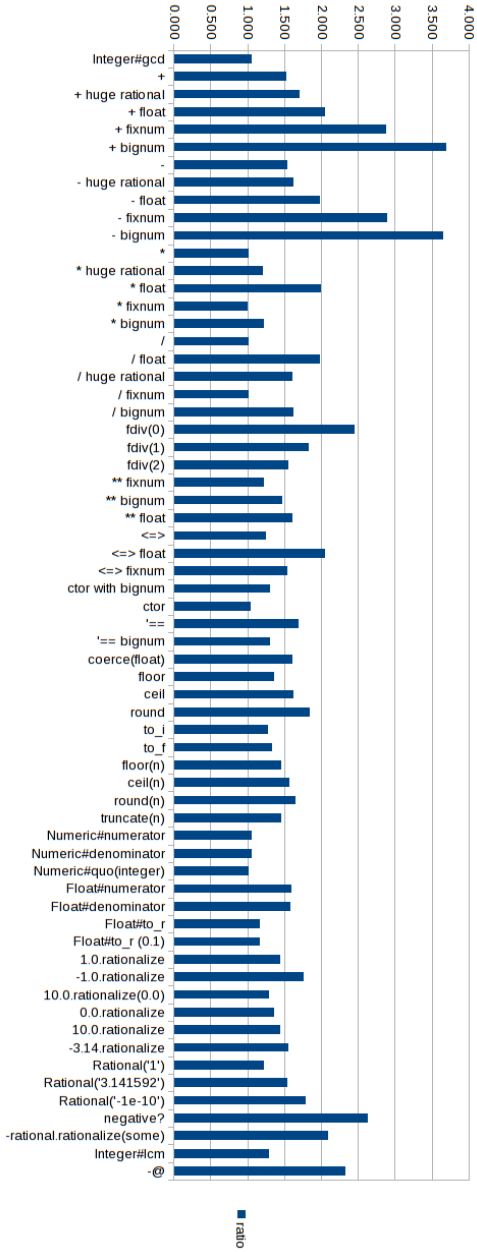
パフォーマンス

パフォーマンスの測定は、資料 1 として添付したスクリプト benchmark.rb で

行った。これは以下のようなものである。

1. 測定対象の処理が一定時間(ソース内では 1 秒で決め打ち)の間に何回繰り返せるかを測る
2. 1 を一定回数(今回は 5 回とした)繰り返し、その中で最も多い回数繰り返せた物をその処理の評価スコアとする

ベンチマークスクリプトで算出したスコアについて、今回作業したバージョンの CRuby が「何倍速になったか」を示したグラフが以下である。ここでは基準として、2016 年 3 月 1 日付の CRuby trunk(revision 53976)を比較対象とした。



グラフから、ほとんど場合でオリジナルの CRuby のパフォーマンスを上回っており、また極端なパフォーマンスの低下部分もない事が分かる。

最も高速になった例と低速だった例を以下に挙げる。

- 最も高速になったのは `Rational + Bignum` のケース。3.7 倍速程度に高速化した。
- 最も低速になったのは `Rational * Fixnum` のケース。0.8%程度低速化した。これについては今回の手法で最適化を施す余地がなかったためコードが変わっておらず、計測誤差の範囲内であると思われる。

実際に計測された数値については、別ページに資料 2 として添付した。

正しさ

実装の正しさの指標として、CRuby 付属の Rational 向けユニットテストを用いた。具体的には以下のファイル群を用いた。(なおパスは CRuby レポジトリ直下を基準としている)

- test/ruby/test_rational.rb
- test/ruby/test_rational2.rb
- test/-ext-/rational/test_rat.rb

これらの全ファイルについて、以下のように失敗もエラーもなく正常にパスする事が確認できた。

```
$ for i in ../test/ruby/test_rational.rb ../test/ruby/test_rational2.rb
../test/-ext-/rational/test_rat.rb; do
>   LD_LIBRARY_PATH=. RUBYLIB=.:../lib:.ext/x86_64-linux:../test/lib  ./ruby-
ragrant $i
> done
Run options:

# Running tests:

Finished tests in 0.016551s, 2718.8203 tests/s, 30692.4606 assertions/s.
45 tests, 508 assertions, 0 failures, 0 errors, 0 skips

ruby -v: ruby 2.4.0dev (2016-03-01 ragrant2015dev.. 53976) [x86_64-linux]
Run options:

# Running tests:

Finished tests in 0.008469s, 118.0752 tests/s, 82888.8096 assertions/s.
1 tests, 702 assertions, 0 failures, 0 errors, 0 skips

ruby -v: ruby 2.4.0dev (2016-03-01 ragrant2015dev.. 53976) [x86_64-linux]
Run options:

# Running tests:

Finished tests in 0.007712s, 388.9972 tests/s, 94785.6606 assertions/s.
3 tests, 731 assertions, 0 failures, 0 errors, 0 skips

ruby -v: ruby 2.4.0dev (2016-03-01 ragrant2015dev.. 53976) [x86_64-linux]
```

ただ残念ながら、付属のほぼすべてのユニットテストを実行する `make test-all` コマンドでは、Segmentation fault を伴う 4 つの失敗が発生してしまった。これは明らかに最適化実装の不具合であるため、上流への成果還元のためには修正が急務である。

成果

3 月 1 日付の CRuby trunk(revision 53976)に対するパッチとして、成果を作成した。全内容は別ページに資料 3 として添付した。

まとめと今後の課題

当初の目論見どおりに最適化が可能なメソッドの全てに置いて、最適化を施すことができた。これは Rational 全メソッド中のほとんどに及び、全面的な高速化を実現できたと言える。

直接的に Rational にまつわるユニットテストもパスし、基本的な振る舞いの正しさを実証できた。ただしより複合的なテストは未だパスしておらず、当初の目標に掲げた「上流への成果の還元」を行うためにはその修正が早急に必要となっている。

急ぎ修正した上、パッチを整理し、さらに論点をまとめた文章を作成した上、ruby-core メーリングリストにてメールで今回の成果の採用を要望する予定である。

また今回の作業では残念ながら未着手となったが、VM レベルの最適化命令として Rational の処理を盛り込めば、メソッド呼び出しのオーバーヘッドが減り、パフォーマンスが向上することが期待される。

最適化作業の確認のためにいくつか取ったプロファイルの例では、malloc()/free()関数が処理の上位に連なる事があった。これはつまり、メモリの確保・開放がパフォーマンスの低下を招いている事を意味している。C レベルでのデータ構造を改良し、即値埋め込みなどを用いることで、さらなるパフォーマンスの改善が望めると考える。

さらに Rational の枠からは外れるが、作業者が CRuby のソースを読む中で気付いた応用例として、Complex の最適化がある。その実装 complex.c の中には今回取り組んだものと同様の問題点が多く見受けられるため、同様の改善手法が有効であると推測する。

資料 1 ベンチマークスクリプト benchmark.rb

```
require 'timeout'
require 'bigdecimal'

$VERBOSE = nil
M = 5
TIMEOUT = 1.0

def measure(proc)
  (1..M).map do
    n = 0
     Timeout.timeout(TIMEOUT) do
      loop do
        proc.()
        n += 1
      end
    end rescue nil
    n
  end.max
end

r = Rational(1, 3)
b1, b2 = (1 << 64) + 1, (1 << 64) + 3
r2 = Rational(b1)
r3 = -r
f, f2 = 10.0, -3.14
i = 10
bd = BigDecimal('0.1')
[
  ['Integer#gcd', ->{b1.gcd(b2)}],
  ['+', ->{r + r}],
  ['+ huge rational', ->{r + r2}],
  ['+ float', ->{r + f}],
  ['+ fixnum', ->{r + i}],
  ['+ bignum', ->{r + b1}],
  ['-', ->{r - r}],
  ['- huge rational', ->{r - r2}],
  ['- float', ->{r - f}],
  ['- fixnum', ->{r - i}],
  ['- bignum', ->{r - b1}],
  ['*', ->{r * r}],
  ['* huge rational', ->{r * r2}],
  ['* float', ->{r * f}],
  ['* fixnum', ->{r * i}],
  ['* bignum', ->{r * b1}],
  ['/', ->{r / r}],
  ['/ float', ->{r / f}],
  ['/ huge rational', ->{r / r2}],
  ['/ fixnum', ->{r / i}],
  ['/ bignum', ->{r / b1}],
  ['fdiv(0)', ->{r.fdiv(0)}],
```

```
  ['fdiv(1)', ->{r.fdiv(1)}],
  ['fdiv(2)', ->{r.fdiv(2)}],
  ['** fixnum', ->{r ** i}],
  ['** bignum', ->{r ** b1}],
  ['** float', ->{r ** f}],
  ['<=>', ->{r <=> r}],
  ['<=> float', ->{r <=> f}],
  ['<=> fixnum', ->{r <=> i}],
  ['ctor with bignum', ->{Rational(1, b1)}],
  ['ctor', ->{Rational(1, 3)}],
  ['==', ->{r == r2}],
  ['== bignum', ->{r2 == b1}],
  ['coerce(float)', ->{r.coerce(f)}],
  ['floor', ->{r.floor}],
  ['ceil', ->{r.ceil}],
  ['round', ->{r.round}],
  ['to_i', ->{r.to_i}],
  ['to_f', ->{r.to_f}],
  ['floor(n)', ->{r.floor(1)}],
  ['ceil(n)', ->{r.ceil(1)}],
  ['round(n)', ->{r.round(1)}],
  ['truncate(n)', ->{r.truncate(1)}],
  ['Numeric#numerator', ->{bd.numerator}],
  ['Numeric#denominator', ->{bd.denominator}],
  ['Numeric#quo(integer)', ->{2.quo(3)}],
  ['Float#numerator', ->{f.numerator}],
  ['Float#denominator', ->{f.denominator}],
  ['Float#to_r', ->{f.to_r}],
  ['Float#to_r (0.1)', ->{0.1.to_r}],
  ['1.0.rationalize', ->{1.0.rationalize}],
  ['-1.0.rationalize', ->{-1.0.rationalize}],
  ['10.0.rationalize(0.0)', ->{f.rationalize(0.0)}],
  ['0.0.rationalize', ->{0.0.rationalize}],
  ['10.0.rationalize', ->{f.rationalize}],
  ['-3.14.rationalize', ->{f2.rationalize}],
  ['Rational(\'1\')', ->{Rational('1')}],
  ['Rational(\'3.141592\')', ->{Rational('3.141592')}],
  ['Rational(\'-1e-10\')', ->{Rational('-1e-10')}],
  ['negative?', ->{r.negative?}],
  ['-rational.rationalize(some)', ->{r3.rationalize(r)}],
  ['Integer#lcm', ->{3.lcm(7)}],
  ['-@', ->{-r}],
].each do |label, proc|
  puts label + "\t%d" % measure(proc)
  STDOUT.flush
end
```

資料 2 ベンチマークの実装値

item	original	optimized	ratio
Integer#gcd	2073654	2156229	1.040
+	2597297	3929714	1.513
+ huge rational	958427	1623212	1.694
+ float	3796303	7733593	2.037
+ fixnum	2204538	6321669	2.868
+ bignum	960551	3535870	3.681
-	2700974	4149679	1.536
- huge rational	862593	1391575	1.613
- float	3895450	7706420	1.978
- fixnum	2215900	6394818	2.886
- bignum	862180	3139436	3.641
*	4416571	4431316	1.003
* huge rational	1444161	1730318	1.198
* float	3812208	7557865	1.983
* fixnum	4733935	4697927	0.992
* bignum	1444558	1747132	1.209
/	4784000	4784947	1.000
/ float	3840098	7555812	1.968
/ huge rational	1114184	1783009	1.600
/ fixnum	4444118	4462815	1.004
/ bignum	1115636	1795752	1.610
fdiv(0)	2505842	6123963	2.444
fdiv(1)	4521609	8222273	1.818
fdiv(2)	2824208	4351012	1.541
** fixnum	2453063	2980055	1.215
** bignum	2439084	3559317	1.459
** float	2655065	4242883	1.598
<=>	6105575	7537532	1.235
<=> float	3696165	7542575	2.041
<=> fixnum	4073961	6207925	1.524
ctor with bignum	1224355	1578876	1.290
ctor	2262095	2342316	1.035
'==	4661646	7808721	1.675
'== bignum	5469109	7087418	1.296
coerce(float)	4155592	6650507	1.600
floor	5770382	7750007	1.343
ceil	4086261	6607083	1.617
round	3167355	5800381	1.831
to_i	6137480	7813317	1.273
to_f	6387082	8490383	1.329
floor(n)	1830352	2644231	1.445
ceil(n)	1584327	2470941	1.560
round(n)	1453313	2386778	1.642
truncate(n)	1819384	2636672	1.449
Numeric#numerator	500092	524130	1.048
Numeric#denominator	499371	523991	1.049
Numeric#quo(integer)	1835170	1843670	1.005
Float#numerator	1961997	3096123	1.578
Float#denominator	1969754	3093764	1.571
Float#to_r	2756661	3185380	1.156
Float#to_r (0.1)	2578294	2969732	1.152
1.0.rationalize	625454	893717	1.429
-1.0.rationalize	445647	779190	1.748
10.0.rationalize(0.0)	1420790	1814648	1.277
0.0.rationalize	2399027	3246796	1.353
10.0.rationalize	602802	860695	1.428
-3.14.rationalize	139655	216264	1.549
Rational('1')	1572380	1909065	1.214
Rational('3.141592')	638795	980339	1.535
Rational('-1e-10')	474266	844058	1.780
negative?	3314818	8704681	2.626
-rational.rationalize(some)	316593	658489	2.080
Integer#lcm	4016243	5150725	1.282
-@	1779816	4117798	2.314

資料 3 作成したパッチ

```
From ef0c3dc797f676aa19d58020d0dfc5b0a7733163 Mon Sep 17 00:00:00 2001
From: Tadashi Saito <tad.a.digger@gmail.com>
Date: Wed, 18 Nov 2015 00:13:18 +0900
Subject: [PATCH] optimize Rational methods.

---
 internal.h |  11 ++
 numeric.c  |  65 +++++-----
 rational.c | 405 ++++++++++++++++++++++++++++++++++++-----------------------
 3 files changed, 264 insertions(+), 217 deletions(-)

diff --git a/internal.h b/internal.h
index dc38623..19a91f2 100644
--- a/internal.h
+++ b/internal.h
@@ -1356,6 +1356,17 @@ int rb_gc_for_fd(int err);

 /* numeric.c (export) */
 VALUE rb_int_positive_pow(long x, unsigned long y);
+VALUE rb_fix_plus(VALUE, VALUE);
+VALUE rb_fix_minus(VALUE, VALUE);
+VALUE rb_fix_mul(VALUE, VALUE);
+VALUE rb_fix_div(VALUE, VALUE);
+VALUE rb_fix_idiv(VALUE, VALUE);
+VALUE rb_fix_fdiv(VALUE, VALUE);
+VALUE rb_fix_modulo(VALUE, VALUE);
+VALUE rb_fix_pow(VALUE, VALUE);
+VALUE rb_fix_cmp(VALUE, VALUE);
+VALUE rb_fix_lshift(VALUE, VALUE);
+VALUE rb_float_pow(VALUE, VALUE);

 /* process.c (export) */
 int rb_exec_async_signal_safe(const struct rb_execarg *e, char *errmsg, size_t errmsg_buflen);
diff --git a/numeric.c b/numeric.c
index c424203..7077080 100644
--- a/numeric.c
+++ b/numeric.c
@@ -104,7 +104,6 @@ round(double x)
 #endif

 static VALUE fix_uminus(VALUE num);
-static VALUE fix_mul(VALUE x, VALUE y);
 static VALUE int_pow(long x, unsigned long y);

 static ID id_coerce, id_div, id_divmod;
@@ -1075,8 +1074,8 @@ flo_divmod(VALUE x, VALUE y)
  *    2.0**3      #=> 8.0
  */

-static VALUE
-flo_pow(VALUE x, VALUE y)
+VALUE
+rb_float_pow(VALUE x, VALUE y)
 {
     double dx, dy;
     if (RB_TYPE_P(y, T_FIXNUM)) {
@@ -2957,8 +2956,8 @@ fix_to_s(int argc, VALUE *argv, VALUE x)
  * +numeric+ and on the magnitude of the result. It may return a Bignum.
  */

-static VALUE
-fix_plus(VALUE x, VALUE y)
+VALUE
+rb_fix_plus(VALUE x, VALUE y)
```

```
 {
     if (FIXNUM_P(y)) {
       long a, b, c;
@@ -2994,8 +2993,8 @@ fix_plus(VALUE x, VALUE y)
  * of +numeric+ and on the magnitude of the result. It may return a Bignum.
  */

-static VALUE
-fix_minus(VALUE x, VALUE y)
+VALUE
+rb_fix_minus(VALUE x, VALUE y)
 {
     if (FIXNUM_P(y)) {
       long a, b, c;
@@ -3033,8 +3032,8 @@ fix_minus(VALUE x, VALUE y)
  * Bignum.
  */

-static VALUE
-fix_mul(VALUE x, VALUE y)
+VALUE
+rb_fix_mul(VALUE x, VALUE y)
 {
     if (FIXNUM_P(y)) {
 #ifdef __HP_cc
@@ -3117,8 +3116,8 @@ fixdivmod(long x, long y, long *divp, long *modp)
  *
  */

-static VALUE
-fix_fdiv(VALUE x, VALUE y)
+VALUE
+rb_fix_fdiv(VALUE x, VALUE y)
 {
     if (FIXNUM_P(y)) {
       return DBL2NUM((double)FIX2LONG(x) / (double)FIX2LONG(y));
@@ -3178,8 +3177,8 @@ fix_divide(VALUE x, VALUE y, ID op)
  * +numeric+ and on the magnitude of the result. It may return a Bignum.
  */

-static VALUE
-fix_div(VALUE x, VALUE y)
+VALUE
+rb_fix_div(VALUE x, VALUE y)
 {
     return fix_divide(x, y, '/');
 }
@@ -3192,8 +3191,8 @@ fix_div(VALUE x, VALUE y)
  * +numeric+.
  */

-static VALUE
-fix_idiv(VALUE x, VALUE y)
+VALUE
+rb_fix_idiv(VALUE x, VALUE y)
 {
     return fix_divide(x, y, id_div);
 }
@@ -3208,8 +3207,8 @@ fix_idiv(VALUE x, VALUE y)
  *  See Numeric#divmod for more information.
  */

-static VALUE
-fix_mod(VALUE x, VALUE y)
+VALUE
+rb_fix_modulo(VALUE x, VALUE y)
 {
     if (FIXNUM_P(y)) {
```

```
       long mod;
@@ -3318,8 +3317,8 @@ rb_int_positive_pow(long x, unsigned long y)
  *    2 ** 0.5    #=> 1.4142135623731
  */

-static VALUE
-fix_pow(VALUE x, VALUE y)
+VALUE
+rb_fix_pow(VALUE x, VALUE y)
 {
     long a = FIX2LONG(x);

@@ -3412,8 +3411,8 @@ fix_equal(VALUE x, VALUE y)
  *  +nil+ is returned if the two values are incomparable.
  */

-static VALUE
-fix_cmp(VALUE x, VALUE y)
+VALUE
+rb_fix_cmp(VALUE x, VALUE y)
 {
     if (x == y) return INT2FIX(0);
     if (FIXNUM_P(y)) {
@@ -3648,7 +3647,7 @@ static VALUE fix_rshift(long, unsigned long);
  * Shifts +fix+ left +count+ positions, or right if +count+ is negative.
  */

-static VALUE
+VALUE
 rb_fix_lshift(VALUE x, VALUE y)
 {
     long val, width;
@@ -4256,23 +4255,23 @@ Init_Numeric(void)
     rb_define_alias(rb_cFixnum, "inspect", "to_s");

     rb_define_method(rb_cFixnum, "-@", fix_uminus, 0);
-    rb_define_method(rb_cFixnum, "+", fix_plus, 1);
-    rb_define_method(rb_cFixnum, "-", fix_minus, 1);
-    rb_define_method(rb_cFixnum, "*", fix_mul, 1);
-    rb_define_method(rb_cFixnum, "/", fix_div, 1);
-    rb_define_method(rb_cFixnum, "div", fix_idiv, 1);
-    rb_define_method(rb_cFixnum, "%", fix_mod, 1);
-    rb_define_method(rb_cFixnum, "modulo", fix_mod, 1);
+    rb_define_method(rb_cFixnum, "+", rb_fix_plus, 1);
+    rb_define_method(rb_cFixnum, "-", rb_fix_minus, 1);
+    rb_define_method(rb_cFixnum, "*", rb_fix_mul, 1);
+    rb_define_method(rb_cFixnum, "/", rb_fix_div, 1);
+    rb_define_method(rb_cFixnum, "div", rb_fix_idiv, 1);
+    rb_define_method(rb_cFixnum, "%", rb_fix_modulo, 1);
+    rb_define_method(rb_cFixnum, "modulo", rb_fix_modulo, 1);
     rb_define_method(rb_cFixnum, "divmod", fix_divmod, 1);
-    rb_define_method(rb_cFixnum, "fdiv", fix_fdiv, 1);
-    rb_define_method(rb_cFixnum, "**", fix_pow, 1);
+    rb_define_method(rb_cFixnum, "fdiv", rb_fix_fdiv, 1);
+    rb_define_method(rb_cFixnum, "**", rb_fix_pow, 1);

     rb_define_method(rb_cFixnum, "abs", fix_abs, 0);
     rb_define_method(rb_cFixnum, "magnitude", fix_abs, 0);

     rb_define_method(rb_cFixnum, "==", fix_equal, 1);
     rb_define_method(rb_cFixnum, "===", fix_equal, 1);
-    rb_define_method(rb_cFixnum, "<=>", fix_cmp, 1);
+    rb_define_method(rb_cFixnum, "<=>", rb_fix_cmp, 1);
     rb_define_method(rb_cFixnum, ">",  fix_gt, 1);
     rb_define_method(rb_cFixnum, ">=", fix_ge, 1);
     rb_define_method(rb_cFixnum, "<",  fix_lt, 1);
@@ -4408,7 +4407,7 @@ Init_Numeric(void)
     rb_define_method(rb_cFloat, "%", flo_mod, 1);
```

```
     rb_define_method(rb_cFloat, "modulo", flo_mod, 1);
     rb_define_method(rb_cFloat, "divmod", flo_divmod, 1);
-    rb_define_method(rb_cFloat, "**", flo_pow, 1);
+    rb_define_method(rb_cFloat, "**", rb_float_pow, 1);
     rb_define_method(rb_cFloat, "==", flo_eq, 1);
     rb_define_method(rb_cFloat, "===", flo_eq, 1);
     rb_define_method(rb_cFloat, "<=>", flo_cmp, 1);
diff --git a/rational.c b/rational.c
index 2359f0e..188a245 100644
--- a/rational.c
+++ b/rational.c
@@ -27,11 +27,27 @@

 #define GMP_GCD_DIGITS 1

+#define INUM_PLUS(x, y) (FIXNUM_P(x) ? rb_fix_plus(x, y) : rb_big_plus(x, y))
+#define INUM_MINUS(x, y) (FIXNUM_P(x) ? rb_fix_minus(x, y) : rb_big_minus(x, y))
+#define INUM_MUL(x, y) (FIXNUM_P(x) ? rb_fix_mul(x, y) : rb_big_mul(x, y))
+#define INUM_DIV(x, y) (FIXNUM_P(x) ? rb_fix_div(x, y) : rb_big_div(x, y))
+#define INUM_IDIV(x, y) (FIXNUM_P(x) ? rb_fix_idiv(x, y) : rb_big_idiv(x, y))
+#define INUM_FDIV(x, y) (FIXNUM_P(x) ? rb_fix_fdiv(x, y) : rb_big_fdiv(x, y))
+#define INUM_MOD(x, y) (FIXNUM_P(x) ? rb_fix_modulo(x, y) : rb_big_modulo(x, y))
+#define INUM_POW(x, y) (FIXNUM_P(x) ? rb_fix_pow(x, y) : rb_big_pow(x, y))
+#define INUM_EQ(x, y) (FIXNUM_P(x) ? f_boolcast(x == y) : rb_big_eq(x, y))
+#define INUM_CMP(x, y) (FIXNUM_P(x) ? rb_fix_cmp(x, y) : rb_big_cmp(x, y))
+#define INUM_LSHIFT(x, y) (FIXNUM_P(x) ? rb_fix_lshift(x, y) : rb_big_lshift(x, y))
+#define INUM_POSITIVE_P(x) (FIXNUM_P(x) ? (FIX2LONG(x) > 0) : !BIGNUM_NEGATIVE_P(x))
+#define INUM_NEGATIVE_P(x) (FIXNUM_P(x) ? (FIX2LONG(x) < 0) : BIGNUM_NEGATIVE_P(x))
+#define INUM_NEGATE(x) (FIXNUM_P(x) ? LONG2NUM(-FIX2LONG(x)) : rb_big_uminus(x))
+#define INUM_ZERO_P(x) (FIXNUM_P(x) ? (FIX2LONG(x) == 0) : rb_bigzero_p(x))
+#define INUM_ABS(x) (INUM_NEGATIVE_P(x) ? INUM_NEGATE(x) : x)
+
 VALUE rb_cRational;

-static ID id_abs, id_cmp, id_convert, id_eqeq_p, id_expt, id_fdiv,
-    id_idiv, id_integer_p, id_negate, id_to_f,
-    id_to_i, id_truncate, id_i_num, id_i_den;
+static ID id_abs, id_eqeq_p, id_idiv, id_integer_p, id_negate, id_to_i,
+    id_i_num, id_i_den;

 #define f_boolcast(x) ((x) ? Qtrue : Qfalse)
 #define f_inspect rb_inspect
@@ -69,24 +85,14 @@ f_add(VALUE x, VALUE y)
 }

 inline static VALUE
-f_cmp(VALUE x, VALUE y)
-{
-    if (FIXNUM_P(x) && FIXNUM_P(y)) {
-      long c = FIX2LONG(x) - FIX2LONG(y);
-      if (c > 0)
-          c = 1;
-      else if (c < 0)
-          c = -1;
-      return INT2FIX(c);
-    }
-    return rb_funcall(x, id_cmp, 1, y);
-}
-
-inline static VALUE
 f_div(VALUE x, VALUE y)
 {
     if (FIXNUM_P(y) && FIX2LONG(y) == 1)
       return x;
+    if (FIXNUM_P(x))
+      return rb_fix_div(x, y);
+    if (RB_TYPE_P(x, T_BIGNUM))
+      return rb_big_div(x, y);
```

```
     return rb_funcall(x, '/', 1, y);
 }

@@ -120,7 +126,10 @@ f_mul(VALUE x, VALUE y)
        }
        else if (ix == 1)
            return y;
-    }
+      else
+          return rb_fix_mul(x, y);
+    } else if (RB_TYPE_P(x, T_BIGNUM))
+      return rb_big_mul(x, y);
     return rb_funcall(x, '*', 1, y);
 }

@@ -132,7 +141,14 @@ f_sub(VALUE x, VALUE y)
     return rb_funcall(x, '-', 1, y);
 }

-fun1(abs)
+inline static VALUE
+f_abs(VALUE x)
+{
+    if (FIXNUM_P(x) || RB_TYPE_P(x, T_BIGNUM))
+      return INUM_ABS(x);
+    return rb_funcall(x, id_abs, 0);
+}
+
 fun1(integer_p)
 fun1(negate)

@@ -143,13 +159,6 @@ f_to_i(VALUE x)
        return rb_str_to_inum(x, 10, 0);
     return rb_funcall(x, id_to_i, 0);
 }
-inline static VALUE
-f_to_f(VALUE x)
-{
-    if (RB_TYPE_P(x, T_STRING))
-      return DBL2NUM(rb_str_to_dbl(x, 0));
-    return rb_funcall(x, id_to_f, 0);
-}

 inline static VALUE
 f_eqeq_p(VALUE x, VALUE y)
@@ -159,21 +168,9 @@ f_eqeq_p(VALUE x, VALUE y)
     return rb_funcall(x, id_eqeq_p, 1, y);
 }

-fun2(expt)
-fun2(fdiv)
 fun2(idiv)

-#define f_expt10(x) f_expt(INT2FIX(10), x)
-
-inline static VALUE
-f_negative_p(VALUE x)
-{
-    if (FIXNUM_P(x))
-      return f_boolcast(FIX2LONG(x) < 0);
-    return rb_funcall(x, '<', 1, ZERO);
-}
-
-#define f_positive_p(x) (!f_negative_p(x))
+#define f_expt10(x) rb_fix_pow(INT2FIX(10), x)

 inline static VALUE
 f_zero_p(VALUE x)
```

```
@@ -327,14 +324,14 @@ f_gcd_normal(VALUE x, VALUE y)
     if (FIXNUM_P(x) && FIXNUM_P(y))
       return LONG2NUM(i_gcd(FIX2LONG(x), FIX2LONG(y)));

-    if (f_negative_p(x))
-      x = f_negate(x);
-    if (f_negative_p(y))
-      y = f_negate(y);
+    if (INUM_NEGATIVE_P(x))
+      x = INUM_NEGATE(x);
+    if (INUM_NEGATIVE_P(y))
+      y = INUM_NEGATE(y);

-    if (f_zero_p(x))
+    if (INUM_ZERO_P(x))
       return y;
-    if (f_zero_p(y))
+    if (INUM_ZERO_P(y))
       return x;

     for (;;) {
@@ -345,7 +342,7 @@ f_gcd_normal(VALUE x, VALUE y)
               return LONG2NUM(i_gcd(FIX2LONG(x), FIX2LONG(y)));
       }
       z = x;
-      x = f_mod(y, x);
+      x = INUM_MOD(y, x);
       y = z;
     }
     /* NOTREACHED */
@@ -389,7 +386,7 @@ f_gcd(VALUE x, VALUE y)
 inline static VALUE
 f_lcm(VALUE x, VALUE y)
 {
-    if (f_zero_p(x) || f_zero_p(y))
+    if (INUM_ZERO_P(x) || INUM_ZERO_P(y))
       return ZERO;
     return f_abs(f_mul(f_div(x, f_gcd(x, y)), y));
 }
@@ -423,8 +420,6 @@ nurat_s_alloc(VALUE klass)
     return nurat_s_new_internal(klass, ZERO, ONE);
 }

-#define rb_raise_zerodiv() rb_raise(rb_eZeroDivError, "divided by 0")
-
 #if 0
 static VALUE
 nurat_s_new_bang(int argc, VALUE *argv, VALUE klass)
@@ -443,13 +438,13 @@ nurat_s_new_bang(int argc, VALUE *argv, VALUE klass)
       if (!k_integer_p(den))
           den = f_to_i(den);

-      switch (FIX2INT(f_cmp(den, ZERO))) {
+      switch (FIX2INT(INUM_CMP(den, ZERO))) {
         case -1:
           num = f_negate(num);
           den = f_negate(den);
           break;
         case 0:
-          rb_raise_zerodiv();
+          rb_num_zerodiv();
           break;
       }
       break;
@@ -483,17 +478,16 @@ inline static void
 nurat_int_check(VALUE num)
 {
     if (!(RB_TYPE_P(num, T_FIXNUM) || RB_TYPE_P(num, T_BIGNUM))) {
```

```
-       if (!k_numeric_p(num) || !f_integer_p(num))
-           rb_raise(rb_eTypeError, "not an integer");
+       rb_raise(rb_eTypeError, "not an Integer");
      }
 }

 inline static VALUE
 nurat_int_value(VALUE num)
 {
-    nurat_int_check(num);
     if (!k_integer_p(num))
       num = f_to_i(num);
+    nurat_int_check(num);
     return num;
 }

@@ -502,13 +496,13 @@ nurat_s_canonicalize_internal(VALUE klass, VALUE num, VALUE den)
 {
     VALUE gcd;

-    switch (FIX2INT(f_cmp(den, ZERO))) {
+    switch (FIX2INT(INUM_CMP(den, ZERO))) {
       case -1:
        num = f_negate(num);
        den = f_negate(den);
        break;
       case 0:
-       rb_raise_zerodiv();
+       rb_num_zerodiv();
        break;
     }

@@ -526,13 +520,13 @@ nurat_s_canonicalize_internal(VALUE klass, VALUE num, VALUE den)
 inline static VALUE
 nurat_s_canonicalize_internal_no_reduce(VALUE klass, VALUE num, VALUE den)
 {
-    switch (FIX2INT(f_cmp(den, ZERO))) {
+    switch (FIX2INT(INUM_CMP(den, ZERO))) {
       case -1:
        num = f_negate(num);
        den = f_negate(den);
        break;
       case 0:
-       rb_raise_zerodiv();
+       rb_num_zerodiv();
        break;
     }

@@ -578,6 +572,7 @@ f_rational_new_no_reduce2(VALUE klass, VALUE x, VALUE y)
     return nurat_s_canonicalize_internal_no_reduce(klass, x, y);
 }

+static VALUE nurat_s_convert(int argc, VALUE *argv, VALUE klass);
 /*
  * call-seq:
  *    Rational(x[, y])  ->  numeric
@@ -608,7 +603,7 @@ f_rational_new_no_reduce2(VALUE klass, VALUE x, VALUE y)
 static VALUE
 nurat_f_rational(int argc, VALUE *argv, VALUE klass)
 {
-    return rb_funcall2(rb_cRational, id_convert, argc, argv);
+    return nurat_s_convert(argc, argv, rb_cRational);
 }

 /*
@@ -648,6 +643,19 @@ nurat_denominator(VALUE self)
     return dat->den;
 }
```

```
+/*
+ * call-seq:
+ *    -rat  ->  rational
+ *
+ * Negates +rat+.
+ */
+static VALUE
+nurat_negate(VALUE self)
+{
+    get_dat1(self);
+    return f_rational_new2(CLASS_OF(self), INUM_NEGATE(dat->num), dat->den);
+}
+
 #ifndef NDEBUG
 #define f_imul f_imul_orig
 #endif
@@ -702,36 +710,37 @@ f_addsub(VALUE self, VALUE anum, VALUE aden, VALUE bnum, VALUE bden, int k)
       VALUE c;

       if (k == '+')
-          c = f_add(a, b);
+          c = INUM_PLUS(a, b);
       else
-          c = f_sub(a, b);
+          c = INUM_MINUS(a, b);

-      b = f_idiv(aden, g);
+      b = INUM_IDIV(aden, g);
       g = f_gcd(c, g);
-      num = f_idiv(c, g);
-      a = f_idiv(bden, g);
-      den = f_mul(a, b);
+      num = INUM_IDIV(c, g);
+      a = INUM_IDIV(bden, g);
+      den = INUM_MUL(a, b);
     }
     else {
       VALUE g = f_gcd(aden, bden);
-      VALUE a = f_mul(anum, f_idiv(bden, g));
-      VALUE b = f_mul(bnum, f_idiv(aden, g));
+      VALUE a = INUM_MUL(anum, INUM_IDIV(bden, g));
+      VALUE b = INUM_MUL(bnum, INUM_IDIV(aden, g));
       VALUE c;

       if (k == '+')
-          c = f_add(a, b);
+          c = INUM_PLUS(a, b);
       else
-          c = f_sub(a, b);
+          c = INUM_MINUS(a, b);

-      b = f_idiv(aden, g);
+      b = INUM_IDIV(aden, g);
       g = f_gcd(c, g);
-      num = f_idiv(c, g);
-      a = f_idiv(bden, g);
-      den = f_mul(a, b);
+      num = INUM_IDIV(c, g);
+      a = INUM_IDIV(bden, g);
+      den = INUM_MUL(a, b);
     }
     return f_rational_new_no_reduce2(CLASS_OF(self), num, den);
 }

+static VALUE nurat_to_f(VALUE self);
 /*
  * call-seq:
```

```
  *    rat + numeric  ->  numeric
@@ -751,13 +760,12 @@ nurat_add(VALUE self, VALUE other)
        {
            get_dat1(self);

-           return f_addsub(self,
-                           dat->num, dat->den,
-                           other, ONE, '+');
+           return f_rational_new_no_reduce2(CLASS_OF(self),
+                                            INUM_PLUS(dat->num, INUM_MUL(other, dat->den)), dat-
>den);
        }
      }
      else if (RB_TYPE_P(other, T_FLOAT)) {
-       return f_add(f_to_f(self), other);
+       return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) + RFLOAT_VALUE(other));
      }
      else if (RB_TYPE_P(other, T_RATIONAL)) {
        {
@@ -792,13 +800,12 @@ nurat_sub(VALUE self, VALUE other)
        {
            get_dat1(self);

-           return f_addsub(self,
-                           dat->num, dat->den,
-                           other, ONE, '-');
+           return f_rational_new_no_reduce2(CLASS_OF(self),
+                                            INUM_MINUS(dat->num, INUM_MUL(other, dat->den)), dat-
>den);
        }
      }
      else if (RB_TYPE_P(other, T_FLOAT)) {
-       return f_sub(f_to_f(self), other);
+       return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) - RFLOAT_VALUE(other));
      }
      else if (RB_TYPE_P(other, T_RATIONAL)) {
        {
@@ -822,9 +829,9 @@ f_muldiv(VALUE self, VALUE anum, VALUE aden, VALUE bnum, VALUE bden, int k)
      if (k == '/') {
        VALUE t;

-       if (f_negative_p(bnum)) {
-           anum = f_negate(anum);
-           bnum = f_negate(bnum);
+       if (INUM_NEGATIVE_P(bnum)) {
+           anum = INUM_NEGATE(anum);
+           bnum = INUM_NEGATE(bnum);
        }
        t = bnum;
        bnum = bden;
@@ -847,8 +854,8 @@ f_muldiv(VALUE self, VALUE anum, VALUE aden, VALUE bnum, VALUE bden, int k)
        VALUE g1 = f_gcd(anum, bden);
        VALUE g2 = f_gcd(aden, bnum);

-       num = f_mul(f_idiv(anum, g1), f_idiv(bnum, g2));
-       den = f_mul(f_idiv(aden, g2), f_idiv(bden, g1));
+       num = INUM_MUL(INUM_IDIV(anum, g1), INUM_IDIV(bnum, g2));
+       den = INUM_MUL(INUM_IDIV(aden, g2), INUM_IDIV(bden, g1));
      }
      return f_rational_new_no_reduce2(CLASS_OF(self), num, den);
  }
@@ -878,7 +885,7 @@ nurat_mul(VALUE self, VALUE other)
        }
      }
      else if (RB_TYPE_P(other, T_FLOAT)) {
-       return f_mul(f_to_f(self), other);
+       return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) * RFLOAT_VALUE(other));
      }
```

```
     else if (RB_TYPE_P(other, T_RATIONAL)) {
       {
@@ -912,7 +919,7 @@ nurat_div(VALUE self, VALUE other)
 {
     if (RB_TYPE_P(other, T_FIXNUM) || RB_TYPE_P(other, T_BIGNUM)) {
       if (f_zero_p(other))
-          rb_raise_zerodiv();
+          rb_num_zerodiv();
       {
           get_dat1(self);

@@ -922,10 +929,10 @@ nurat_div(VALUE self, VALUE other)
       }
     }
     else if (RB_TYPE_P(other, T_FLOAT))
-      return rb_funcall(f_to_f(self), '/', 1, other);
+      return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) / RFLOAT_VALUE(other));
     else if (RB_TYPE_P(other, T_RATIONAL)) {
       if (f_zero_p(other))
-          rb_raise_zerodiv();
+          rb_num_zerodiv();
       {
           get_dat2(self, other);

@@ -956,9 +963,17 @@ nurat_div(VALUE self, VALUE other)
 static VALUE
 nurat_fdiv(VALUE self, VALUE other)
 {
+    VALUE div;
     if (f_zero_p(other))
-      return f_div(self, f_to_f(other));
-    return f_to_f(f_div(self, other));
+      return DBL2NUM(RFLOAT_VALUE(nurat_to_f(self)) / 0.0);
+    if (FIXNUM_P(other) && FIX2LONG(other) == 1)
+      return nurat_to_f(self);
+    div = nurat_div(self, other);
+    if (RB_TYPE_P(div, T_RATIONAL))
+      return nurat_to_f(div);
+    if (RB_TYPE_P(div, T_FLOAT))
+      return div;
+    return rb_funcall(div, rb_intern("to_f"), 0);
 }

 inline static VALUE
@@ -1007,8 +1022,8 @@ nurat_expt(VALUE self, VALUE other)
               return f_rational_new_bang1(CLASS_OF(self), INT2FIX(f_odd_p(other) ? -1 : 1));
           }
           else if (f_zero_p(dat->num)) {
-              if (FIX2INT(f_cmp(other, ZERO)) == -1) {
-                  rb_raise_zerodiv();
+              if (FIX2INT(rb_fix_cmp(ZERO, other)) == 1) {
+                  rb_num_zerodiv();
               }
               else {
                   return f_rational_new_bang1(CLASS_OF(self), ZERO);
@@ -1024,14 +1039,14 @@ nurat_expt(VALUE self, VALUE other)

           get_dat1(self);

-          switch (FIX2INT(f_cmp(other, ZERO))) {
+          switch (FIX2INT(rb_fix_cmp(other, ZERO))) {
             case 1:
-              num = f_expt(dat->num, other);
-              den = f_expt(dat->den, other);
+              num = INUM_POW(dat->num, other);
+              den = INUM_POW(dat->den, other);
               break;
             case -1:
```

```
-              num = f_expt(dat->den, f_negate(other));
-              den = f_expt(dat->num, f_negate(other));
+              num = INUM_POW(dat->den, INUM_NEGATE(other));
+              den = INUM_POW(dat->num, INUM_NEGATE(other));
               break;
             default:
               num = ONE;
@@ -1043,13 +1058,13 @@ nurat_expt(VALUE self, VALUE other)
     }
     else if (RB_TYPE_P(other, T_BIGNUM)) {
       rb_warn("in a**b, b may be too big");
-      return f_expt(f_to_f(self), other);
+      return rb_float_pow(nurat_to_f(self), other);
     }
     else if (RB_TYPE_P(other, T_FLOAT) || RB_TYPE_P(other, T_RATIONAL)) {
-      return f_expt(f_to_f(self), other);
+      return rb_float_pow(nurat_to_f(self), other);
     }
     else {
-      return rb_num_coerce_bin(self, other, id_expt);
+      return rb_num_coerce_bin(self, other, rb_intern("**"));
     }
 }

@@ -1075,12 +1090,12 @@ nurat_cmp(VALUE self, VALUE other)
           get_dat1(self);

           if (FIXNUM_P(dat->den) && FIX2LONG(dat->den) == 1)
-              return f_cmp(dat->num, other); /* c14n */
-          return f_cmp(self, f_rational_new_bang1(CLASS_OF(self), other));
+              return INUM_CMP(dat->num, other); /* c14n */
+          other = f_rational_new_bang1(CLASS_OF(self), other);
       }
     }
-    else if (RB_TYPE_P(other, T_FLOAT)) {
-      return f_cmp(f_to_f(self), other);
+    if (RB_TYPE_P(other, T_FLOAT)) {
+      return rb_dbl_cmp(RFLOAT_VALUE(nurat_to_f(self)), RFLOAT_VALUE(other));
     }
     else if (RB_TYPE_P(other, T_RATIONAL)) {
       {
@@ -1094,14 +1109,14 @@ nurat_cmp(VALUE self, VALUE other)
               num2 = f_imul(FIX2LONG(bdat->num), FIX2LONG(adat->den));
           }
           else {
-              num1 = f_mul(adat->num, bdat->den);
-              num2 = f_mul(bdat->num, adat->den);
+              num1 = INUM_MUL(adat->num, bdat->den);
+              num2 = INUM_MUL(bdat->num, adat->den);
           }
-          return f_cmp(f_sub(num1, num2), ZERO);
+          return INUM_CMP(INUM_MINUS(num1, num2), ZERO);
       }
     }
     else {
-      return rb_num_coerce_cmp(self, other, id_cmp);
+      return rb_num_coerce_cmp(self, other, rb_intern("<=>"));
     }
 }

@@ -1124,30 +1139,29 @@ nurat_eqeq_p(VALUE self, VALUE other)
       {
           get_dat1(self);

-          if (f_zero_p(dat->num) && f_zero_p(other))
+          if (INUM_ZERO_P(dat->num) && INUM_ZERO_P(other))
               return Qtrue;
```

```
            if (!FIXNUM_P(dat->den))
                return Qfalse;
            if (FIX2LONG(dat->den) != 1)
                return Qfalse;
-           if (f_eqeq_p(dat->num, other))
-               return Qtrue;
-           return Qfalse;
+           return INUM_EQ(dat->num, other);
        }
      }
      else if (RB_TYPE_P(other, T_FLOAT)) {
-       return f_eqeq_p(f_to_f(self), other);
+       return f_boolcast(rb_dbl_cmp(RFLOAT_VALUE(nurat_to_f(self)), RFLOAT_VALUE(other))
+                         == INT2FIX(0));
      }
      else if (RB_TYPE_P(other, T_RATIONAL)) {
        {
            get_dat2(self, other);

-           if (f_zero_p(adat->num) && f_zero_p(bdat->num))
+           if (INUM_ZERO_P(adat->num) && INUM_ZERO_P(bdat->num))
                return Qtrue;

-           return f_boolcast(f_eqeq_p(adat->num, bdat->num) &&
-                             f_eqeq_p(adat->den, bdat->den));
+           return f_boolcast(INUM_EQ(adat->num, bdat->num) &&
+                             INUM_EQ(adat->den, bdat->den));
        }
      }
      else {
@@ -1163,7 +1177,7 @@ nurat_coerce(VALUE self, VALUE other)
        return rb_assoc_new(f_rational_new_bang1(CLASS_OF(self), other), self);
      }
      else if (RB_TYPE_P(other, T_FLOAT)) {
-       return rb_assoc_new(other, f_to_f(self));
+       return rb_assoc_new(other, nurat_to_f(self));
      }
      else if (RB_TYPE_P(other, T_RATIONAL)) {
        return rb_assoc_new(other, self);
@@ -1214,18 +1228,44 @@ nurat_true(VALUE self)
 }
 #endif

+/*
+ *  call-seq:
+ *     rat.positive? ->  true or false
+ *
+ *  Returns +true+ if +rat+ is greater than 0.
+ */
+static VALUE
+nurat_positive_p(VALUE self)
+{
+    get_dat1(self);
+    return f_boolcast(INUM_POSITIVE_P(dat->num));
+}
+
+/*
+ *  call-seq:
+ *     rat.negative? ->  true or false
+ *
+ *  Returns +true+ if +rat+ is less than 0.
+ */
+static VALUE
+nurat_negative_p(VALUE self)
+{
+    get_dat1(self);
+    return f_boolcast(INUM_NEGATIVE_P(dat->num));
+}
```

```
+
 static VALUE
 nurat_floor(VALUE self)
 {
     get_dat1(self);
-    return f_idiv(dat->num, dat->den);
+    return INUM_IDIV(dat->num, dat->den);
 }

 static VALUE
 nurat_ceil(VALUE self)
 {
     get_dat1(self);
-    return f_negate(f_idiv(f_negate(dat->num), dat->den));
+    return INUM_NEGATE(INUM_IDIV(INUM_NEGATE(dat->num), dat->den));
 }

 /*
@@ -1247,9 +1287,9 @@ static VALUE
 nurat_truncate(VALUE self)
 {
     get_dat1(self);
-    if (f_negative_p(dat->num))
-      return f_negate(f_idiv(f_negate(dat->num), dat->den));
-    return f_idiv(dat->num, dat->den);
+    if (INUM_NEGATIVE_P(dat->num))
+      return INUM_NEGATE(INUM_IDIV(INUM_NEGATE(dat->num), dat->den));
+    return INUM_IDIV(dat->num, dat->den);
 }

 static VALUE
@@ -1261,17 +1301,17 @@ nurat_round(VALUE self)

     num = dat->num;
     den = dat->den;
-    neg = f_negative_p(num);
+    neg = INUM_NEGATIVE_P(num);

     if (neg)
-      num = f_negate(num);
+      num = INUM_NEGATE(num);

-    num = f_add(f_mul(num, TWO), den);
-    den = f_mul(den, TWO);
-    num = f_idiv(num, den);
+    num = INUM_PLUS(INUM_MUL(num, TWO), den);
+    den = INUM_MUL(den, TWO);
+    num = INUM_IDIV(num, den);

     if (neg)
-      num = f_negate(num);
+      num = INUM_NEGATE(num);

     return num;
 }
@@ -1290,10 +1330,10 @@ f_round_common(int argc, VALUE *argv, VALUE self, VALUE (*func)(VALUE))
       rb_raise(rb_eTypeError, "not an integer");

     b = f_expt10(n);
-    s = f_mul(self, b);
+    s = nurat_mul(self, b);

     if (k_float_p(s)) {
-      if (f_lt_p(n, ZERO))
+      if (INUM_NEGATIVE_P(n))
         return ZERO;
       return self;
     }
```

```
@@ -1304,10 +1344,10 @@ f_round_common(int argc, VALUE *argv, VALUE self, VALUE (*func)(VALUE))

     s = (*func)(s);

-    s = f_div(f_rational_new_bang1(CLASS_OF(self), s), b);
+    s = nurat_div(f_rational_new_bang1(CLASS_OF(self), s), b);

-    if (f_lt_p(n, ONE))
-      s = f_to_i(s);
+    if (FIX2INT(INUM_CMP(n, ONE)) < 0)
+      s = nurat_truncate(s);

     return s;
 }
@@ -1424,7 +1464,7 @@ static VALUE
 nurat_to_f(VALUE self)
 {
     get_dat1(self);
-    return f_fdiv(dat->num, dat->den);
+    return INUM_FDIV(dat->num, dat->den);
 }

 /*
@@ -1560,8 +1600,8 @@ nurat_rationalize(int argc, VALUE *argv, VALUE self)
     if (argc == 0)
       return self;

-    if (f_negative_p(self))
-      return f_negate(nurat_rationalize(argc, argv, f_abs(self)));
+    if (nurat_negative_p(self))
+      return nurat_negate(nurat_rationalize(argc, argv, nurat_negate(self)));

     rb_scan_args(argc, argv, "01", &e);
     e = f_abs(e);
@@ -1684,7 +1724,7 @@ nurat_marshal_load(VALUE self, VALUE a)
     if (RARRAY_LEN(a) != 2)
       rb_raise(rb_eArgError, "marshaled rational must have an array whose length is 2 but %ld",
RARRAY_LEN(a));
     if (f_zero_p(RARRAY_AREF(a, 1)))
-      rb_raise_zerodiv();
+      rb_num_zerodiv();

     rb_ivar_set(self, id_i_num, RARRAY_AREF(a, 0));
     rb_ivar_set(self, id_i_den, RARRAY_AREF(a, 1));
@@ -1766,8 +1806,6 @@ rb_rational_new(VALUE x, VALUE y)
     return nurat_s_canonicalize_internal(rb_cRational, x, y);
 }

-static VALUE nurat_s_convert(int argc, VALUE *argv, VALUE klass);
-
 VALUE
 rb_Rational(VALUE x, VALUE y)
 {
@@ -1807,7 +1845,7 @@ rb_rational_den(VALUE rat)
 static VALUE
 numeric_numerator(VALUE self)
 {
-    return f_numerator(f_to_r(self));
+    return nurat_numerator(f_to_r(self));
 }

 /*
@@ -1819,7 +1857,7 @@ numeric_numerator(VALUE self)
 static VALUE
 numeric_denominator(VALUE self)
 {
-    return f_denominator(f_to_r(self));
+    return nurat_denominator(f_to_r(self));
```

```
 }


@@ -1835,7 +1873,7 @@ static VALUE
 numeric_quo(VALUE x, VALUE y)
 {
     if (RB_TYPE_P(y, T_FLOAT)) {
-        return f_fdiv(x, y);
+        return rb_funcall(x, rb_intern("fdiv"), 1, y);
     }

 #ifdef CANON
@@ -1847,7 +1885,7 @@ numeric_quo(VALUE x, VALUE y)
         x = rb_convert_type(x, T_RATIONAL, "Rational", "to_r");
     }
-    return rb_funcall(x, '/', 1, y);
+    return nurat_div(x, y);
 }


@@ -1875,6 +1913,7 @@ integer_denominator(VALUE self)
     return INT2FIX(1);
 }

+static VALUE float_to_r(VALUE self);
 /*
  * call-seq:
  *    flo.numerator  ->  integer
@@ -1891,7 +1930,7 @@ float_numerator(VALUE self)
     double d = RFLOAT_VALUE(self);
     if (isinf(d) || isnan(d))
       return self;
-    return rb_call_super(0, 0);
+    return nurat_numerator(float_to_r(self));
 }

 /*
@@ -1909,7 +1948,7 @@ float_denominator(VALUE self)
     double d = RFLOAT_VALUE(self);
     if (isinf(d) || isnan(d))
       return INT2FIX(1);
-    return rb_call_super(0, 0);
+    return nurat_denominator(float_to_r(self));
 }

 /*
@@ -1991,9 +2030,6 @@ float_decode(VALUE self)
 }
 #endif

-#define id_lshift rb_intern("<<")
-#define f_lshift(x,n) rb_funcall((x), id_lshift, 1, (n))
-
 /*
  * call-seq:
  *    flt.to_r  ->  rational
@@ -2021,14 +2057,17 @@ float_to_r(VALUE self)
       long ln = FIX2LONG(n);

       if (ln == 0)
-          return f_to_r(f);
+          return rb_rational_new1(f);
       if (ln > 0)
-          return f_to_r(f_lshift(f, n));
+          return rb_rational_new1(INUM_LSHIFT(f, n));
       ln = -ln;
-      return rb_rational_new2(f, f_lshift(ONE, INT2FIX(ln)));
```

```
+       return rb_rational_new2(f, rb_fix_lshift(ONE, INT2FIX(ln)));
      }
 #else
-    return f_to_r(f_mul(f, f_expt(INT2FIX(FLT_RADIX), n)));
+    f = INUM_MUL(f, rb_fix_pow(INT2FIX(FLT_RADIX), n));
+    if (RB_TYPE_P(f, T_RATIONAL))
+      return f;
+    return rb_rational_new1(f);
 #endif
 }

@@ -2042,7 +2081,7 @@ rb_flt_rationalize_with_prec(VALUE flt, VALUE prec)
     b = f_add(flt, e);

     if (f_eqeq_p(a, b))
-        return f_to_r(flt);
+        return float_to_r(flt);

     nurat_rationalize_internal(a, b, &p, &q);
     return rb_rational_new2(p, q);
@@ -2054,33 +2093,33 @@ rb_flt_rationalize(VALUE flt)
     VALUE a, b, f, n, p, q;

     float_decode_internal(flt, &f, &n);
-    if (f_zero_p(f) || f_positive_p(n))
-        return rb_rational_new1(f_lshift(f, n));
+    if (INUM_ZERO_P(f) || FIX2INT(n) >= 0)
+        return rb_rational_new1(INUM_LSHIFT(f, n));

 #if FLT_RADIX == 2
     {
         VALUE two_times_f, den;

-        two_times_f = f_mul(TWO, f);
-        den = f_lshift(ONE, f_sub(ONE, n));
+        two_times_f = rb_fix_mul(TWO, f);
+        den = rb_fix_lshift(ONE, rb_fix_minus(ONE, n));

-        a = rb_rational_new2(f_sub(two_times_f, ONE), den);
-        b = rb_rational_new2(f_add(two_times_f, ONE), den);
+        a = rb_rational_new2(INUM_MINUS(two_times_f, ONE), den);
+        b = rb_rational_new2(INUM_PLUS(two_times_f, ONE), den);
     }
 #else
     {
         VALUE radix_times_f, den;

-        radix_times_f = f_mul(INT2FIX(FLT_RADIX), f);
-        den = f_expt(INT2FIX(FLT_RADIX), f_sub(ONE, n));
+        radix_times_f = rb_fix_mul(INT2FIX(FLT_RADIX), f);
+        den = rb_fix_pow(INT2FIX(FLT_RADIX), rb_fix_minus(ONE, n));

-        a = rb_rational_new2(f_sub(radix_times_f, INT2FIX(FLT_RADIX - 1)), den);
-        b = rb_rational_new2(f_add(radix_times_f, INT2FIX(FLT_RADIX - 1)), den);
+        a = rb_rational_new2(INUM_MINUS(radix_times_f, INT2FIX(FLT_RADIX - 1)), den);
+        b = rb_rational_new2(INUM_PLUS(radix_times_f, INT2FIX(FLT_RADIX - 1)), den);
     }
 #endif

-    if (f_eqeq_p(a, b))
-        return f_to_r(flt);
+    if (nurat_eqeq_p(a, b))
+        return float_to_r(flt);

     nurat_rationalize_internal(a, b, &p, &q);
     return rb_rational_new2(p, q);
@@ -2104,9 +2143,10 @@ static VALUE
 float_rationalize(int argc, VALUE *argv, VALUE self)
```

```
 {
     VALUE e;
+    double d = RFLOAT_VALUE(self);

-    if (f_negative_p(self))
-        return f_negate(float_rationalize(argc, argv, f_abs(self)));
+    if (d < 0.0)
+        return nurat_negate(float_rationalize(argc, argv, DBL2NUM(-d)));

     rb_scan_args(argc, argv, "01", &e);

@@ -2217,9 +2257,9 @@ read_num(const char **s, int numsign, int strict,
             return 0;
       {
             VALUE l = f_expt10(INT2NUM(count));
-           *num = f_mul(*num, l);
-           *num = f_add(*num, fp);
-           *num = f_div(*num, l);
+           *num = nurat_mul(*num, l);
+           *num = nurat_add(*num, fp);
+           *num = nurat_div(*num, l);
       }
     }

@@ -2231,14 +2271,14 @@ read_num(const char **s, int numsign, int strict,
       if (!read_digits(s, strict, &exp, NULL))
           return 0;
       if (expsign == '-')
-          exp = f_negate(exp);
+          exp = INUM_NEGATE(exp);
     }

     if (numsign == '-')
-      *num = f_negate(*num);
+      *num = nurat_negate(*num);
     if (!NIL_P(exp)) {
       VALUE l = f_expt10(exp);
-      *num = f_mul(*num, l);
+      *num = nurat_mul(*num, l);
     }
     return 1;
 }
@@ -2265,7 +2305,7 @@ read_rat_nos(const char **s, int sign, int strict,
       if (!read_den(s, strict, &den))
           return 0;
       if (!(FIXNUM_P(den) && FIX2LONG(den) == 1))
-          *num = f_div(*num, den);
+          *num = nurat_div(*num, den);
     }
     return 1;
 }
@@ -2421,14 +2461,14 @@ nurat_s_convert(int argc, VALUE *argv, VALUE klass)
     rb_match_busy(backref);

     if (RB_TYPE_P(a1, T_FLOAT)) {
-      a1 = f_to_r(a1);
+      a1 = float_to_r(a1);
     }
     else if (RB_TYPE_P(a1, T_STRING)) {
       a1 = string_to_r_strict(a1);
     }

     if (RB_TYPE_P(a2, T_FLOAT)) {
-      a2 = f_to_r(a2);
+      a2 = float_to_r(a2);
     }
     else if (RB_TYPE_P(a2, T_STRING)) {
       a2 = string_to_r_strict(a2);
```

```
@@ -2510,17 +2550,11 @@ Init_Rational(void)
     assert(fprintf(stderr, "assert() is now active\n"));

     id_abs = rb_intern("abs");
-    id_cmp = rb_intern("<=>");
-    id_convert = rb_intern("convert");
     id_eqeq_p = rb_intern("==");
-    id_expt = rb_intern("**");
-    id_fdiv = rb_intern("fdiv");
     id_idiv = rb_intern("div");
     id_integer_p = rb_intern("integer?");
     id_negate = rb_intern("-@");
-    id_to_f = rb_intern("to_f");
     id_to_i = rb_intern("to_i");
-    id_truncate = rb_intern("truncate");
     id_i_num = rb_intern("@numerator");
     id_i_den = rb_intern("@denominator");

@@ -2541,6 +2575,7 @@ Init_Rational(void)
     rb_define_method(rb_cRational, "numerator", nurat_numerator, 0);
     rb_define_method(rb_cRational, "denominator", nurat_denominator, 0);

+    rb_define_method(rb_cRational, "-@", nurat_negate, 0);
     rb_define_method(rb_cRational, "+", nurat_add, 1);
     rb_define_method(rb_cRational, "-", nurat_sub, 1);
     rb_define_method(rb_cRational, "*", nurat_mul, 1);
@@ -2562,6 +2597,8 @@ Init_Rational(void)
     rb_define_method(rb_cRational, "rational?", nurat_true, 0);
     rb_define_method(rb_cRational, "exact?", nurat_true, 0);
 #endif
+    rb_define_method(rb_cRational, "positive?", nurat_positive_p, 0);
+    rb_define_method(rb_cRational, "negative?", nurat_negative_p, 0);

     rb_define_method(rb_cRational, "floor", nurat_floor_n, -1);
     rb_define_method(rb_cRational, "ceil", nurat_ceil_n, -1);
--
2.7.0
```