Partial Sage Implementation of First Part of Class Group Algorithm from Stevenhagen's Paper

William Stein

February 2, 2009

Listing 1: Primes up to a bound

```
def primes_of_bounded_norm(K, B):
    """
    Return sorted list of the primes up to a given norm B.
    EXAMPLES:
        sage: K.<a> = NumberField(x^2 + 5)
        sage: primes_of_bounded_norm(K, 3)
        [Fractional ideal (2, a + 1), Fractional ideal (3, a + 1),
         Fractional ideal (3, a - 1)]
    """
    # Find all prime ideals of norm up to B by
    # factoring all primes of ZZ of norm up to B, then
    # taken the primes above them.
    v = []
    for p in primes(B+1):   # B+1 to include endpoint
        for P in K.primes_above(p):
            if P.norm() <= B:
                v.append(P)
    return v
```

Listing 2: Computing the valuation map F

```
def valuation_map(K, T):
    """
    INPUT:
        K -- number field
        T -- set of primes
    OUTPUT:
        a function from T-units in K to ZZ^#T.
        Raises a ValueError if the input isn't a T-unit.

    EXAMPLES:
        sage: K.<a> = NumberField(x^2 + 5)
        sage: v = primes_of_bounded_norm(K,3)
        sage: F = valuation_map(K, v)
        sage: F(2)
        (2, 0, 0)
        sage: F(3)
```

```
        (0, 1, 1)
        sage: F(5)
        Traceback (most recent call last):
        ...
        ValueError: u (=5) is not a T-unit
    """
    ZZT = ZZ^len(T)
    S = set(T)
    def F(u):
        fac = dict(K.factor(u))
        # Make sure u is a T-unit
        if not set(fac.keys()).issubset(S):
            raise ValueError, "u (=%s) is not a T-unit"%u
        return ZZT([fac[P] if fac.has_key(P) else 0 for P in T])
    return F
```

Listing 3: Invariants of the class group

```
def class_group(f, B=None, k_bound=None, verbose=True):
    """
    INPUT:
        f -- monic irreducible poly over ZZ
        B -- factor basis bound (default: min of Bach and Minkowski)
        k_bound -- search for smooth values of f(k) for |k| < k_bound
                   (default: 100)
    OUTPUT: computation of group that covers Cl(K) using Stevenhagen Algorithm
    """
    # normalize types and check preconditions
    f = f.polynomial(ZZ)
    if not f.is_monic(): raise ValueError, "f must be monic"

    # Make the number field
    K.<alpha> = NumberField(f)
    if verbose: print "Pari gives class number = ", K.class_number()

    # Compute bound if necessary
    if B is None:
        B = floor(min(K.minkowski_bound(), 12*log(abs(K.discriminant()))^2))
    if B <= 1:
        print "Obviously trivial class group"
        return [], None
    if verbose: print "B  = ", B

    # Compute the factor basis T using this B
    T = primes_of_bounded_norm(K, B)
    if verbose: print "#T = ", len(T)

    # Compute the map F from the group of T-units
    # to the free abelian group on T, which we view
    # as #T copies of ZZ.
    F = valuation_map(K, T)
```

```
    # Compute the T-units coming from primes of ZZ with support in T
    units_ZZ = []
    primes_ZZ = prime_range(B+1)
    for p in primes_ZZ:
        try:
            units_ZZ.append((K(p), F(p)))
        except ValueError: pass

    # Compute smooth f(k)
    if k_bound is None: k_bound = 100

    primes_ZZ_set = set(primes_ZZ)
    units_smooth = []
    for k in [-k_bound..k_bound]:
        w = f(k)
        if set(w.prime_divisors()).issubset(primes_ZZ_set):
            try:
                units_smooth.append((alpha-k, F(alpha-k)))
            except ValueError: pass

    # Now we have all the units we're going to get from *this* algorithm.
    units = units_ZZ + units_smooth

    # Create relation matrix
    A = matrix([u[1] for u in units])

    # Elementary divisors gives the structure of the covering of the
    # class group
    e = A.elementary_divisors()[:A.ncols()]
    e = [i for i in e if i != 1]
    return e, A
    # Also, we get generators for a subgroup of the unit group from ker(A)...
    # But we ignore that for now.
```

Listing 4: Some easy small examples

```
sage: e, A = class_group(x^2 -7); e, A.nrows()
Pari gives class number = 1
B = 2
#T = 1
([], 3)
sage: e, A = class_group(x^3 - 2); e, A.nrows()
Pari gives class number = 1
B = 2
#T = 1
([], 3)
sage: e, A = class_group(x^2 + 23); e, A.nrows()
Pari gives class number = 3
B = 3
#T = 4
([3], 26)
sage: e, A = class_group(x^3 + 44, verbose=True); e, A.nrows()
```

```
Pari gives class number =  1
B  =  10
#T =  4
([], 8)
sage: e, A = class_group(x^3 + x^2 + 5*x - 16); e, A.nrows()
Pari gives class number =  4
B  =  26
#T =  12
([4], 26)
```

Listing 5: Examples where the method doesn't work

```
sage: e, A = class_group(x^4 - 2*x^2 + 3*x - 7); e, A.nrows()
Pari gives class number =  1
B  =  37
#T =  14
([0], 13)
sage: e, A = class_group(x^4 - 2*x^2 + 3*x - 7, k_bound=1000); e, A.nrows()
Pari gives class number =  1
B  =  37
#T =  14
([0], 13)
sage: e, A = class_group(x^4 - 2*x^2 + 3*x - 7, k_bound=1000, B=100); e, A.nrows()
Pari gives class number =  1
B  =  100
#T =  28
([0], 23)
sage: e, A = class_group(x^4 + 3*x - 15); e, A.nrows()
Pari gives class number =  1
B  =  111
#T =  32
([2, 0, 0], 23)
sage: e, A = class_group(x^4 + 3*x - 15, k_bound=1000, B=200); e, A.nrows()
Pari gives class number =  1
B  =  200
#T =  51
([2, 0, 0], 36)
```

Listing 6: Cyclotomic Fields

```
sage: e, A = class_group(SR(cyclotomic_polynomial(7))); e, A.nrows()
Pari gives class number =  1; B  =  4; #T =  0; ([], 2)
sage: e, A = class_group(SR(cyclotomic_polynomial(11))); e, A.nrows()
Pari gives class number =  1; B  =  58; #T =  11; ([0, 0, 0], 5)
sage: e, A = class_group(SR(cyclotomic_polynomial(11)), k_bound=500); e, A.nrows()
Pari gives class number =  1; B  =  58; #T =  11; ([0, 0, 0], 5)
```