3.3.2 Manin symbols

As above, fix coset representatives $r_0, \ldots, r_m$ for $\Gamma_0(N)$ in $\mathrm{SL}_2(\mathbb{Z})$. Consider formal symbols $[r_i]'$ for $i = 0, \ldots, m$. Let $[r_i]$ be the modular symbol $r_i\{0, \infty\} = \{r_i(0), r_i(\infty)\}$. We equip the symbols $[r_0]', \ldots, [r_m]'$ with a right action of $\mathrm{SL}_2(\mathbb{Z})$, which is given by $[r_i]'.g = [r_j]'$, where $\Gamma_0(N)r_j = \Gamma_0(N)r_i g$. We extend the notation by writing $[\gamma]' = [\Gamma_0(N)\gamma]' = [r_i]'$, where $\gamma \in \Gamma_0(N)r_i$. Then the right action of $\Gamma_0(N)$ is simply $[\gamma]'.g = [\gamma g]'$.

Theorem 1.1.2 implies that $\mathrm{SL}_2(\mathbb{Z})$ is generated by the two matrices $\sigma = \left(\begin{smallmatrix} 0 & -1 \\ 1 & 0 \end{smallmatrix}\right)$ and $\tau = \left(\begin{smallmatrix} 1 & -1 \\ 1 & 0 \end{smallmatrix}\right)$. Note that $\sigma = S$ from Theorem 1.1.2 and $\tau = TS$, so $T = \tau\sigma \in \langle \sigma, \tau \rangle$.

The following theorem provides us with a finite presentation for the space $\mathcal{M}_2(\Gamma_0(N))$ of modular symbols.

Theorem 3.3.4 (Manin)**.** *Consider the quotient M of the free abelian group on Manin symbols $[r_0]', \ldots, [r_m]'$ modulo the subgroup generated by the elements (for all i):*

$$[r_i]' + [r_i]'\sigma \qquad \textit{and} \qquad [r_i]' + [r_i]'\tau + [r_i]'\tau^2,$$

and modulo any torsion. Then there is an isomorphism $\Psi : M \xrightarrow{\sim} \mathbb{M}_2(\Gamma_0(N))$ given by $[r_i]' \mapsto [r_i] = r_i\{0, \infty\}$.

Proof. We will only prove that Ψ is surjective; the proof that Ψ is injective requires much more work and will be omitted from this book (see [Man72, §1.7] for a complete proof). [[**Todo: And reference my book with Ribet, or Wiese's work?**]]

Proposition 3.3.2 implies that Ψ is surjective, assuming that Ψ is well defined. We next verify that Ψ is well defined, i.e. that the listed two and three term relations hold in the image. To see that the first relation holds, note that

$$\begin{aligned} [r_i] + [r_i]\sigma &= \{r_i(0), r_i(\infty)\} + \{r_i\sigma(0), r_i\sigma(\infty)\} \\ &= \{r_i(0), r_i(\infty)\} + \{r_i(\infty), r_i(0)\} \\ &= 0. \end{aligned}$$

For the second relation we have

$$\begin{aligned} [r_i] + [r_i]\tau + [r_i]\tau^2 &= \{r_i(0), r_i(\infty)\} + \{r_i\tau(0), r_i\tau(\infty)\} + \{r_i\tau^2(0), r_i\tau^2(\infty)\} \\ &= \{r_i(0), r_i(\infty)\} + \{r_i(\infty), r_i(1)\} + \{r_i(1), r_i(0)\} \\ &= 0 \end{aligned}$$

□

Example 3.3.5. By default SAGE computes modular symbols spaces over $\mathbb{Q}$, i.e., $\mathbb{M}_2(\Gamma_0(N); \mathbb{Q}) \cong \mathbb{M}_2(\Gamma_0(N)) \otimes \mathbb{Q}$. SAGE represents (weight 2) Manin symbols as pairs (c, d). Here c, d are integers that satisfy $0 \le c, d < N$; they define a point $(c : d) \in \mathbb{P}^1(\mathbb{Z}/N\mathbb{Z})$, hence a right coset of $\Gamma_0(N)$ in $\mathrm{SL}_2(\mathbb{Z})$ (see Proposition 3.3.1).

Create $\mathbb{M}_2(\Gamma_0(N);\mathbb{Q})$ in SAGE by typing `ModularSymbols(N, 2)`. We then use the SAGE command `manin_generators` to enumerate a list of generators $[r_0],\ldots,[r_n]$ as in Theorem 3.3.4 for several spaces of modular symbols.

```
sage: M = ModularSymbols(2,2)
sage: M
Full Modular Symbols space for Gamma_0(2) of weight 2 with
sign 0 and dimension 1 over Rational Field
sage: M.manin_generators()
[(0,1), (1,0), (1,1)]

sage: M = ModularSymbols(3,2)
sage: M.manin_generators()
[(0,1), (1,0), (1,1), (1,2)]

sage: M = ModularSymbols(6,2)
sage: M.manin_generators()
[(0,1), (1,0), (1,1), (1,2), (1,3), (1,4), (1,5), (2,1),
 (2,3), (2,5), (3,1), (3,2)]
```

Given `x=(c,d)`, the command `x.lift_to_sl2z(N)` finds an element `[a,b,c',d']` of $\mathrm{SL}_2(\mathbb{Z})$ whose lower two entries are congruent to (c,d) modulo N.

```
sage: M = ModularSymbols(2,2)
sage: [x.lift_to_sl2z(2) for x in M.manin_generators()]
[[1, 0, 0, 1], [0, -1, 1, 0], [0, -1, 1, 1]]
sage: M = ModularSymbols(6,2)
sage: x = M.manin_generators()[9]
sage: x
(2,5)
sage: x.lift_to_sl2z(6)
[1, 2, 2, 5]
```

The `manin_basis` command returns a list of indices into the Manin generator list such that the corresponding symbols form a basis for the quotient of the $\mathbb{Q}$-vector space spanned by Manin symbols modulo the 2 and 3-term relations of Theorem 3.3.4.

```
sage: M = ModularSymbols(2,2)
sage: M.manin_basis()
[1]
sage: [M.manin_generators()[i] for i in M.manin_basis()]
[(1,0)]
sage: M = ModularSymbols(6,2)
sage: M.manin_basis()
[1, 10, 11]
sage: [M.manin_generators()[i] for i in M.manin_basis()]
[(1,0), (3,1), (3,2)]
```

Thus, e.g., every element of $\mathbb{M}_2(\Gamma_0(6))$ is a $\mathbb{Q}$-linear combination of the 3 symbols $[(1,0)]$, $[(3,1)]$, and $[(3,2)]$. We can write each of these as a modular symbol using the `modular_symbol_rep` function.

```
sage: M.basis()
((1,0), (3,1), (3,2))
sage: [x.modular_symbol_rep() for x in M.basis()]
[{Infinity,0}, {0,1/3}, {-1/2,-1/3}]
```

The `manin_gens_to_basis` function returns a matrix whose rows express each Manin symbol generator in terms of the subset of Manin symbols that forms a basis (as returned by `manin_basis`.

```
sage: M = ModularSymbols(2,2)
sage: M.manin_gens_to_basis()
[-1]
[ 1]
[ 0]
```

Since the basis is $(1,0)$ this means that in $\mathbb{M}_2(\Gamma_0(2);\mathbb{Q})$, we have $[(0,1)] = -[(1,0)]$ and $[(1,1)] = 0$. (Since no denominators are involved, we have in fact computed a presentation of $\mathbb{M}_2(\Gamma_0(2);\mathbb{Z})$.)

Convert a Manin symbol $x = (c,d)$ to an element of a modular symbols space M, use `M(xx)`:

```
sage: M = ModularSymbols(2,2)
sage: x = (1,0); M(x)
(1,0)
sage: M( (3,1) )    # entries are reduced modulo $2$ first
0
sage: M( (10,19) )
-(1,0)
```

Next consider $\mathbb{M}_2(\Gamma_0(6);\mathbb{Q})$:

```
sage: M = ModularSymbols(6,2)
sage: M.manin_gens_to_basis()
[-1  0  0]
[ 1  0  0]
[ 0  0  0]
[ 0 -1  1]
[ 0 -1  0]
[ 0 -1  1]
[ 0  0  0]
[ 0  1 -1]
[ 0  0 -1]
[ 0  1 -1]
[ 0  1  0]
[ 0  0  1]
```

Recalling that our choice of basis for $\mathbb{M}_2(\Gamma_0(6);\mathbb{Q})$ is $[(1,0)],[(3,1)],[(3,2)]$. Thus, e.g., first row of this matrix says that $[(0,1)] = -[(1,0)]$, and the fourth row asserts that $[(1,2)] = -[(3,1)] + [(3,2)]$.

```
sage: M = ModularSymbols(6,2)
sage: M((0,1))
-(1,0)
sage: M((1,2))
-(3,1) + (3,2)
```

3.4 Hecke Operators

3.4.1 Hecke Operators on Modular Symbols

When p is a prime not dividing N, define

$$T_p\{\alpha,\beta\} = \begin{pmatrix} p & 0 \\ 0 & 1 \end{pmatrix}\{\alpha,\beta\} + \sum_{r \bmod p} \begin{pmatrix} 1 & r \\ 0 & p \end{pmatrix}\{\alpha,\beta\}.$$

As mentioned before, this definition is compatible with the integration pairing $\langle\, , \rangle$ of Section 3.1, in the sense that $\langle fT_p, x\rangle = \langle f, T_p x\rangle$. When $p \mid N$, the definition is the same, except that the matrix $\left(\begin{smallmatrix} p & 0 \\ 0 & 1 \end{smallmatrix}\right)$ is not included in the sum. (There is a similar definition of T_n for n composite; see Section 8.3.1 for the general definition.)

Example 3.4.1. For example, when $N = 11$ we have

$$\begin{aligned} T_2\{0,1/5\} &= \{0,2/5\} + \{0,1/10\} + \{1/2,3/5\} \\ &= -2\{0,1/5\}. \end{aligned}$$

3.4.2 Hecke Operators on Manin Symbols

In [Mer94], L. Merel gives a description of the action of T_p directly on Manin symbols $[r_i]$ (see Section 8.3.2 for details). For example, when $p = 2$ and N is odd, we have

$$T_2([r_i]) = [r_i]\begin{pmatrix}1 & 0\\ 0 & 2\end{pmatrix} + [r_i]\begin{pmatrix}2 & 0\\ 0 & 1\end{pmatrix} + [r_i]\begin{pmatrix}2 & 1\\ 0 & 1\end{pmatrix} + [r_i]\begin{pmatrix}1 & 0\\ 1 & 2\end{pmatrix}. \quad (3.4.1)$$

For any prime, let S_p be the set of matrices constructed using the following algorithm (see [Cre97a, §2.4]):

Algorithm 3.4.2 (Cremona's Matrices)**.** *Given a prime p, this algorithm outputs a list of 2×2 matrices of determinant p that can be used to compute the Hecke operator T_p.*

1. Output $\begin{pmatrix}1 & 0\\ 0 & p\end{pmatrix}$.
2. For $r = \left\lceil -\frac{p}{2} \right\rceil, \ldots, \left\lfloor \frac{p}{2} \right\rfloor$:
   (a) Let $x_1 = p$, $x_2 = -r$, $y_1 = 0$, $y_2 = 1$, $a = -p$, $b = r$.
   (b) Output $\begin{pmatrix}x_1 & x_2\\ y_1 & y_2\end{pmatrix}$.
   (c) As long as $b \neq 0$, do the following:
      i. Let q be the integer closest to a/b (if a/b is a half integer round away from 0).
      ii. Let $c = a - bq$, $a = -b$, $b = c$.
      iii. Set $x_3 = qx_2 - x_1$, $x_1 = x_2$, $x_2 = x_3$, and $y_3 = qy_2 - y_1$, $y_1 = y_2$, $y_2 = y_3$,
      iv. Output $\begin{pmatrix}x_1 & x_2\\ y_1 & y_2\end{pmatrix}$.

Proposition 3.4.3 (Cremona, Merel)**.** *Let S_p be as above. Then for $p \nmid N$ and $[x] \in \mathcal{M}_2(\Gamma_0(N))$ a Manin symbol, we have*

$$T_p([x]) = \sum_{g \in S_p} [xg].$$

Proof. See Proposition 2.4.1 of [Cre97a]. □

There are other lists of matrices, due to Merel, that work even when $p \mid N$ (see Section 8.3.2).

The command `HeilbronnCremonaList(p)`, for p prime, gives a list of matrices that computes T_p on Manin symbols for $p \nmid N$.

```
sage: HeilbronnCremonaList(2)
[[1, 0, 0, 2], [2, 0, 0, 1], [2, 1, 0, 1], [1, 0, 1, 2]]
sage: HeilbronnCremonaList(3)
[[1, 0, 0, 3], [3, 1, 0, 1], [1, 0, 1, 3], [3, 0, 0, 1],
 [3, -1, 0, 1], [-1, 0, 1, -3]]
sage: HeilbronnCremonaList(5)
[[1, 0, 0, 5], [5, 2, 0, 1], [2, 1, 1, 3], [1, 0, 3, 5],
 [5, 1, 0, 1], [1, 0, 1, 5], [5, 0, 0, 1], [5, -1, 0, 1],
 [-1, 0, 1, -5], [5, -2, 0, 1], [-2, 1, 1, -3], [1, 0, -3, 5]]
sage: len(HeilbronnCremonaList(97))
392
```

Example 3.4.4. Using SAGE we compute the matrix of T_2 on $\mathbb{M}_2(\Gamma_0(2))$:

```
sage: M = ModularSymbols(2,2)
sage: M.T(2).matrix()
[1]
```

Example 3.4.5. We use SAGE to compute Hecke operators on $\mathbb{M}_2(\Gamma_0(6))$:

```
sage: M = ModularSymbols(6, 2)
sage: M.T(2).matrix()
[ 2  1 -1]
[-1  0  1]
[-1 -1  2]
sage: M.T(3).matrix()
[3 2 0]
[0 1 0]
[2 2 1]
```

In fact for $p \geq 5$ we have $T_p = p + 1$, since $M_2(\Gamma_0(6))$ is spanned by generalized Eisenstein series (see Chapter 5).

Example 3.4.6. We use SAGE to compute Hecke operators on $\mathbb{M}_2(\Gamma_0(39))$:

```
sage: M = ModularSymbols(39, 2)
sage: T2 = M.T(2)
sage: T2.matrix()
[ 3  0 -1  0  0  1  1 -1  0]
[ 0  0  2  0 -1  1  0  1 -1]
[ 0  1  0 -1  1  1  0  1 -1]
[ 0  0  1  0  0  1  0  1 -1]
[ 0 -1  2  0  0  1  0  1 -1]
[ 0  0  1  1  0  1  1 -1  0]
[ 0  0  0 -1  0  1  1  2  0]
[ 0  0  0  1  0  0  2  0  1]
[ 0  0 -1  0  0  0  1  0  2]
sage: factor(T2.charpoly())
(x - 3)^3 * (x - 1)^2 * (x^2 + 2*x - 1)^2
```

Notice that the Hecke operators commute, so their eigenspace structure is similar.

```
sage: T2 = M.T(2).matrix()
sage: T5 = M.T(5).matrix()
sage: T2*T5 - T5*T2 == 0
True
sage: T5.charpoly().factor()
(x - 6)^3 * (x - 2)^2 * (x^2 - 8)^2
```

The rational decomposition of T_2 is a list of the kernels of $(f^e)(T_2)$, where f runs through the irreducible factors of the characteristic polynomial of T_2 and f^e exactly divides this characteristic polynomial. Using SAGE we find them:

```
sage: M = ModularSymbols(39, 2)
sage: M.T(2).decomposition()
[Dimension 3 subspace of a modular symbols space of level 39,
 Dimension 2 subspace of a modular symbols space of level 39,
 Dimension 4 subspace of a modular symbols space of level 39]
```

3.5 Computing the boundary map

In Section 3.2 we defined a map $\mathbb{M}_2(\Gamma_0(N)) \to \mathbb{B}_2(\Gamma_0(N))$ whose kernel $\mathbb{S}_2(\Gamma_0(N))$ is called the space of cuspidal modular symbols. This kernel will be important in computing cuspforms in Section 3.7 below.

To compute the boundary map on Manin symbols, note that $[\gamma] = \{\gamma(0), \gamma(\infty)\}$, so if $\gamma = \left(\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right)$, then

$$\delta([\gamma]) = \{\gamma(\infty)\} - \{\gamma(0)\} = \{a/c\} - \{b/d\}.$$

Computing this boundary map would appear to first require an algorithm to compute the set $C(\Gamma_0(N)) = \Gamma_0(N)\backslash\mathbb{P}^1(\mathbb{Q})$ of cusps for $\Gamma_0(N)$. In fact, there is a trick to compute the set of cusps in the course of running the algorithm. First, give an algorithm for deciding whether or not two elements of $\mathbb{P}^1(\mathbb{Q})$ are equivalent modulo the action of $\Gamma_0(N)$. Then simply construct $C(\Gamma_0(N))$ in the course of computing the boundary map, i.e., keep a list of cusps found so far, and whenever a new cusp class is discovered add it to the list. The following proposition, which is proved in [Cre97a, Prop. 2.2.3], explains how to determine whether two cusps are equivalent.

Proposition 3.5.1 (Cremona). *Let (c_i, d_i), $i = 1, 2$ be pairs of integers with $\gcd(c_i, d_i) = 1$, and possibly $d_i = 0$. There exists $g \in \Gamma_0(N)$ such that $g(c_1/d_1) = c_2/d_2$ in $\mathbb{P}^1(\mathbb{Q})$ if and only if*

$$s_1 d_2 \equiv s_2 d_1 \pmod{\gcd(d_1 d_2, N)}$$

where s_j satisfies $c_j s_j \equiv 1 \pmod{d_j}$.

In SAGE the command `boundary_map()` computes the boundary map from $\mathbb{M}_2(\Gamma_0(N))$ to $\mathbb{B}_2(\Gamma_0(N))$, and the `cuspidal_submodule()` command computes its kernel. For example, for level 2 the boundary map is given by the matrix $[1\ \ -1]$, and its kernel is the 0 space.

```
sage: M = ModularSymbols(2, 2)
sage: M.boundary_map()
Hecke module morphism boundary map defined by the matrix
[ 1 -1]
Domain: Full Modular Symbols space for Gamma_0(2) of weight 2 with sign ...
Codomain: Space of Boundary Modular Symbols for Gamma0(2) of weight 2 and ...
sage: M.cuspidal_submodule()
Dimension 0 subspace of a modular symbols space of level 2
```

The smallest level for which the boundary map has nontrivial kernel, i.e., for which $\mathbb{S}_2(\Gamma_0(N)) \neq 0$ is $N = 11$.

```
sage: M = ModularSymbols(11, 2)
sage: M.boundary_map().matrix()
[ 1 -1]
[ 0  0]
[ 0  0]
sage: M.cuspidal_submodule()
Dimension 2 subspace of a modular symbols space of level 11
sage: S = M.cuspidal_submodule(); S
Dimension 2 subspace of a modular symbols space of level 11
sage: S.basis()
((1,8), (1,9))
```

The following illustrates that the Hecke operators preserve $\mathbb{S}_2(\Gamma_0(N))$:

```
sage: S.T(2).matrix()
[-2  0]
[ 0 -2]
sage: S.T(3).matrix()
[-1  0]
[ 0 -1]
sage: S.T(5).matrix()
[1 0]
[0 1]
```

A nontrivial fact (the Eichler-Shimura relation, etc.) is that for p prime the eigenvalue of each of these matrices is the same as $p+1-\#E(\mathbb{F}_p)$, where E is the elliptic curve $X_0(11)$ given by the equation

$$y^2 + y = x^3 - x^2 - 10x - 20.$$

```
sage: E = EllipticCurve([0,-1,1,-10,-20])
sage: 2 + 1 - E.Np(2)
-2
sage: 3 + 1 - E.Np(3)
-1
sage: 5 + 1 - E.Np(5)
1
sage: print [S.T(p).matrix()[0,0] - (p+1-E.Np(p)) for p in primes(100)]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```