LOADING BELYI COVERS IN MAGMA

ARSEN ELKIN

In order to use the database, you will first need to download auxiliary files `util.m` and `format.m`.

The former contains utilities for working with Belyi passports, the latter – MAGMA records describing Belyi maps on the projective line, elliptic and hyperelliptic curves. This document describes the process for loading data into MAGMA, as well as routines for manipulating this data.

1. What are Belyi maps

According to a theorem of Belyi[1], an algebraic complex curve can be defined over $\overline{\mathbb{Q}}$ if and only if there exists a regular map from it to the projective line $\mathbb{P}^1$ branched over only three points. By composing such a map with an appropriate fractional linear transformation, we may assume that these points are at ∞, 0, and 1. Such a rational function is called a Belyi map.

To every Belyi map, we associate its *passport*, which is a list of ramification indices at points lying over ∞, 0, and 1. For example, Belyi cover $\varphi : \mathbb{P}^1 \to \mathbb{P}^1$ given on the affine part of $\mathbb{P}^1$ by $\varphi(x) = -3x^4 + 4x^3$ is ramified over ∞ at ∞, over 0 at 0 and $-4/3$, and over 1 at 1 and $(1 \pm \sqrt{-2})/3$, with respective ramification indices 4, 3, 1, 2, 1 and 1, and therefore has passport $[[4], [3, 1], [2, 1, 1]]$, which we shorten to $4, 31, 211$.

In our datatbase, the lists of ramification indices over ∞, 0, and 1 are ordered in reverse lexicographic order. To obtain a map with an unordered passport, compose with an appropriate automorphism of $\mathbb{P}^1$.

2. Loading database

To load Belyi map data for, say, passport $5, 5, 311$, make sure that files `util.m`, `format.m`, and `belyi-5-5-311.m` are located in the same directory, and, while running MAGMA in that directory, type

```
> load "database.m";
```

This will append records for the maps to the database `BelyiRecords`, which will be created if it does not yet exist.

3. Accessing the database

Access to data stored in the database is accomplished through two methods.

`NumberOfBelyiMaps(DB, passport)`: The number of Belyi maps in a database with a given passport.

`BelyiMap(DB, passport, i)`: The *i*th Belyi map with a given passport.

We demonstrate the usage of these functions with the following example:

```
> load "database.m";
...
> database := BelyiMapDatabase();
> passport1 := [[5],[1,3,1],[5]];
> NumberOfBelyiMaps(database, passport1);
14
> phi1 := BelyiMap(database, passport1, 5);
> phi1;
(1/5*(4*w^3 + 36*w)*x + 1/72*(31*w^3 + 434*w))*y - 5*x^2 + 1/432*(-35*w^2 -
    420)*x + 63425/124416
> BelyiPassportToString(BelyiPassport(phi1));
[[5],[3,1,1],[5]]
> Parent(phi1);
Function Field of Hyperelliptic Curve defined by y^2 = x^3 + 1345/41472*x +
    1/53747712*(237025*w^2 + 2844300) over Number Field with defining polynomial
t^4 + 24*t^2 + 150 over the Rational Field
> curve1 := Curve(Parent(phi1));
> curve1;
Hyperelliptic Curve defined by y^2 = x^3 + 1345/41472*x + 1/53747712*(237025*w^2
    + 2844300) over Number Field with defining polynomial t^4 + 24*t^2 + 150
over the Rational Field
> Genus(curve1);
1
> IsEllipticCurve(curve1);
true Elliptic Curve defined by y^2 = x^3 + 1345/41472*x +
1/53747712*(237025*w^2 + 2844300) over Number Field with defining
polynomial z^4 + 24*z^2 + 150 over the Rational Field
...
> passport2 := [[4,1,1],[2,2,1,1],[6]];
> NumberOfBelyiMaps(database, passport2);
3
> phi2 := BelyiMap(database, passport2, 1);
> phi2;
(2*x^6 - 3*x^4 + 1)/(2*x^6 - 3*x^4)
> BelyiPassportToString(BelyiPassport(phi2));
[[4,1,1],[2,2,1,1],[6]]
> Parent(phi2);
Function Field of Curve over Rational Field defined by
0
> curve2 := Curve(Parent(phi2));
> Genus(curve2);
0
```

4. Internal data formats

Belyi passports. Belyi passports are represented by a sequence of 3 sequences of integer numbers, representing ramification indices of the map over the points $\infty, 0, 1 \in \mathbb{P}^1$, respectively. For example, `[[5], [3,2], [2,2,1]]` is the passport of a map $C \to \mathbb{P}^1$ of degree 5, ramified at a single point over ∞, in two points over 0 with degrees 3 and 2, and in three points over 1 with degrees 2, 2 and 1.

Genus zero. A function on the projective line is just a rational function

$$\varphi = \frac{p(x)}{r(x)}.$$

with $p, r \in K[x]$, where K is a field of definition of the map. Therefore, genus 0 Belyi map record is defined as

```
G0BelyiRecord := recformat<
          passport:   SeqEnum,
          defpoly:    SeqEnum,
          coeffs_p:   SeqEnum,
          coeffs_r:   SeqEnum,
          field:      Fld,
          phi:        FldFunFracSchElt,
          genus:      RngIntElt,
          degree:     RngIntElt
          >;
```

The fields are as follows

- **passport:** the passport.
- **defpoly:** the coefficients of the defining polynomial of the field of definition. This is a list of rational numbers.
- **coeff_p:** the coefficients of the numerator.
- **coeff_r:** the coefficients of the denominator.
- **field:** the field of definition
- **phi:** the map itself.
- **genus:** the genus.
- **degree:** the degree.

Hyperelliptic curves. A hyperelliptic curve C defined over a field K is given by an equation

$$y^2 + h(x)y = f(x),$$

where $f, h \in K[x]$. A rational map on such a curve can then be defined by

$$\varphi = \frac{p(x) + q(x)y}{r(x)}$$

for some polynomials $p, q, r \in K[x]$. As a consequence, the record describing a Belyi map on a hyperelliptic curve is has the following structure:

```
HypBelyiRecord := recformat<
          passport:   SeqEnum,
          defpoly:    SeqEnum,
          coeffs_f:   SeqEnum,
          coeffs_h:   SeqEnum,
          coeffs_p:   SeqEnum,
          coeffs_q:   SeqEnum,
          coeffs_r:   SeqEnum,
          field:      Fld,
          curve:      CrvHyp,
          phi:        FldFunFracSchElt,
          genus:      RngIntElt,
          degree:     RngIntElt
          >;
```

The fields are as follows

- **passport:** the passport.
- **defpoly:** the coefficients of the defining polynomial of the field of definition. This is a list of rational numbers.
- **coeff_f:** the coefficients of the the polynomial f in the above description.
- **coeff_h:** the coefficients of the polynomial h.
- **coeff_p:** the coefficients of the polynomial p.
- **coeff_q:** the coefficients of the polynomial q.
- **coeff_r:** the coefficients of the polynomial r.
- **field:** a field of definition of the curve and the map.
- **curve:** the curve
- **phi:** the map, as an element of the function field of the curve.
- **genus:** the genus.
- **degree:** the degree.

5. Methods in `util.m`

`BelyiDegree(passport)`: Compute the degree of a map corresponding to a given Belyi passport.

`BelyiGenus(passport)`: Compute the genus of a curve with a Belyi map corresponding to a given passport.

`BelyiPassport(f)`: Computes ramification indices over ∞, 0, and 1 of a given rational map. For Belyi maps, this coincides with their passport.

`HasBelyiPassport(f, passport)`: Verifies whether a function has a given Belyi passport. Functionally equivalent to `BelyiPassport(f) eq passport`, but is potentially twice as fast.

`ValidatePassport(passport)`: Check that a given sequence is a valid Belyi passport.

6. Methods in `format.m`

`MakeBelyiRecord(phi)`: Returns the Belyi record of a map.

`ReadBelyiRecord(record)`: Returns the map corresponding to a given Belyi record.

`FillBelyiRecord(~record)`: When a Belyi record is loaded from a file, the fields for the field of definition, map, degree, and genus are not evaluated in order to save memory. This procedure fills in these missing details.

The file `format.m` also provides methods `MakeG0BelyiRecord(phi)`, `ReadG0BelyiRecord(record)`, `FillG0BelyiRecord(~record)`, `MakeHypBelyiRecord(phi)`, `ReadHypBelyiRecord(record)`, `FillHypBelyiRecord(~record)`, which performs the same functions but on records of type `G0BelyiRecord` and `HypBelyiRecord`, respectively.

7. Example

```
> load "format.m";
Loading "format.m"
Loading "utils.m"
> load "belyi-3-3-3.m";
Loading "belyi-3-3-3.m";
> records := BelyiRecords_3_3_3();
> records[7];
rec<HypBelyiRecord |
    passport := [
        [ 3 ],
        [ 3 ],
        [ 3 ]
    ],
    defpoly := [ 1, -1, 1 ],
    coeffs_f := [
        [ 1, 0 ],
        [ 0, 0 ],
        [ 0, 0 ],
        [ 1, 0 ]
    ],
    coeffs_h := [],
    coeffs_p := [
        [ 2, 0 ],
        [ 0, -6 ],
        [ -6, 6 ],
        [ 2, 0 ]
    ],
    coeffs_q := [
        [ 6, 0 ],
        [ 0, -6 ]
    ],
    coeffs_r := [
        [ -8, 0 ],
        [ 0, -12 ],
        [ 6, -6 ],
        [ 1, 0 ]
    ]>
> FillBelyiRecord(~records[7]);
> records[7];
```

```
rec<HypBelyiRecord |
    passport := [
        [ 3 ],
        [ 3 ],
        [ 3 ]
    ],
    defpoly := [ 1, -1, 1 ],
    coeffs_f := [
        [ 1, 0 ],
        [ 0, 0 ],
        [ 0, 0 ],
        [ 1, 0 ]
    ],
    coeffs_h := [],
    coeffs_p := [
        [ 2, 0 ],
        [ 0, -6 ],
        [ -6, 6 ],
        [ 2, 0 ]
    ],
    coeffs_q := [
        [ 6, 0 ],
        [ 0, -6 ]
    ],
    coeffs_r := [
        [ -8, 0 ],
        [ 0, -12 ],
        [ 6, -6 ],
        [ 1, 0 ]
    ],
    field := Number Field with defining polynomial t^2 - t + 1 over
        the Rational Field,
    curve := Hyperelliptic Curve defined by y^2 = x^3 + 1 over Number
        Field with defining polynomial t^2 - t + 1 over the Rational
        Field,
    phi := (-6*w*x + 6)/(x^3 + (-6*w + 6)*x^2 - 12*w*x - 8)*y + (2*x^3
        + (6*w - 6)*x^2 - 6*w*x + 2)/(x^3 + (-6*w + 6)*x^2 - 12*w*x -
        8),
    genus := 1,
    degree := 3>
> // Notice additional fields now appearing in the record.
> ReadBelyiRecord(records[7]);
(-6*w*x + 6)/(x^3 + (-6*w + 6)*x^2 - 12*w*x - 8)*y + (2*x^3 + (6*w -
    6)*x^2 - 6*w*x + 2)/(x^3 + (-6*w + 6)*x^2 - 12*w*x - 8)
```

References

[1] Belyi, G. V. *Galois Extensions of a Maximal Cyclotomic Field*, (Russian) Izv. Akad. Nauk SSSR Ser. Mat. **43** (1979), no. 2, 267–276, 479.

Mathematics Institute, University of Warwick, CV47AL, United Kingdom
E-mail address: `A.Elkin@warwick.ac.uk`