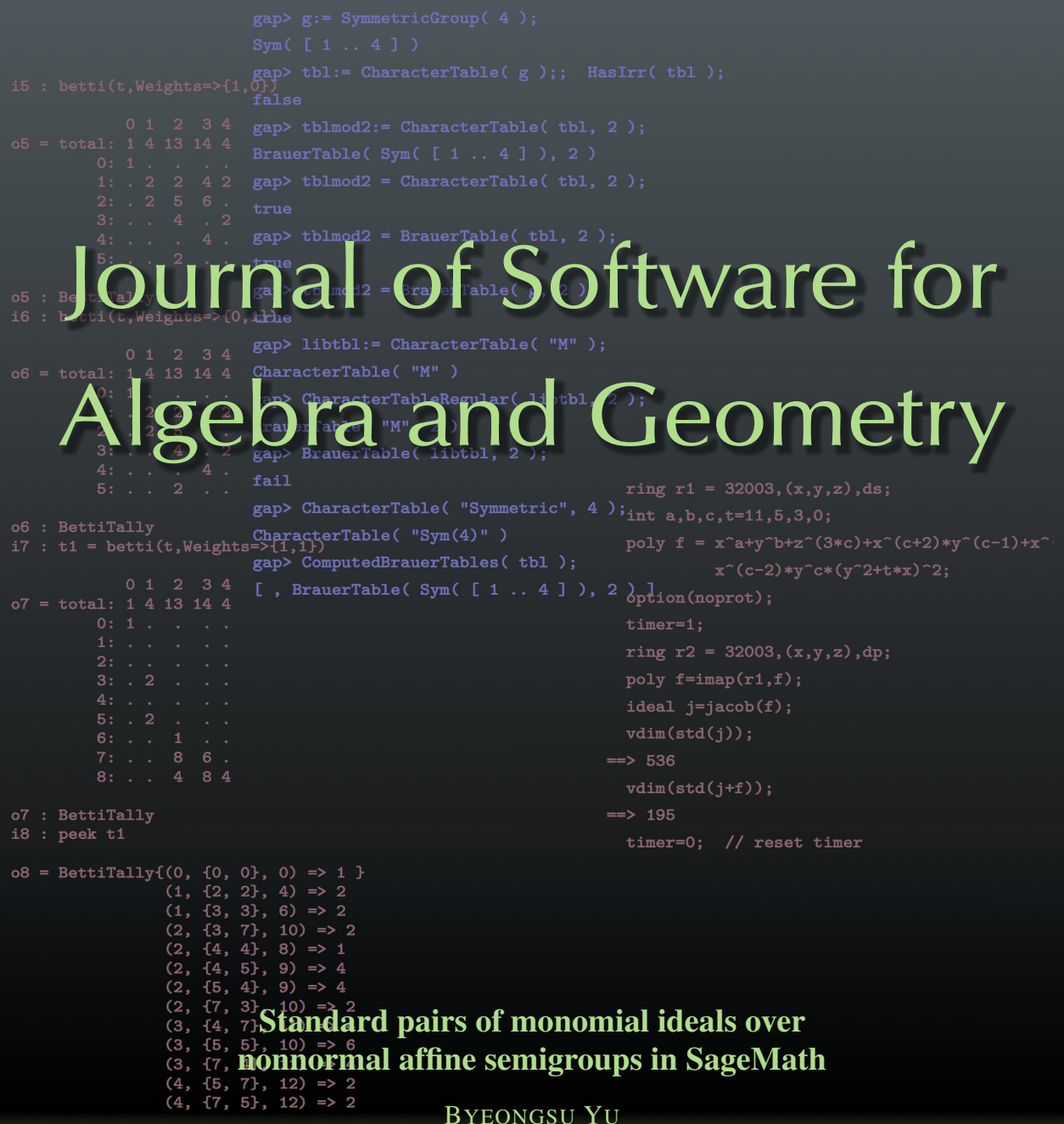

Journal of Software for Algebra and Geometry

Standard pairs of monomial ideals over nonnormal affine semigroups in SageMath

BYEONGSU YU

vol 11 2021

JSAG **11** (2021), 129–142
https://doi.org/10.2140/jsag.2021.11.129

The Journal of Software for
Algebra and Geometry

Standard pairs of monomial ideals over nonnormal affine semigroups in SageMath

BYEONGSU YU

ABSTRACT: We present `StdPairs`, a `SageMath` library to compute standard pairs of a monomial ideal over a pointed (nonnormal) affine semigroup ring. Moreover, `StdPairs` provides the associated prime ideals, the corresponding multiplicities, and an irredundant irreducible primary decomposition of a monomial ideal. The library expands on the `standardPairs` function on `Macaulay2` over polynomial rings, and is based on algorithms from Matusevich and Yu (2020). We also provide methods that allow the outputs from this library to be compatible with the `Normaliz` package of `Macaulay2` and `SageMath`.

1. INTRODUCTION.

Affine semigroup rings are the object of many studies in combinatorial commutative algebra. The goal of this article is to present the `SageMath` library `StdPairs`, which systematizes computations for monomial ideals in affine semigroup rings. The algorithms implemented here are based on the notion of *standard pairs*, introduced for monomial ideals in polynomial rings by [Sturmfels et al. 1995], and generalized to the semigroup ring case in [Matusevich and Yu 2020]. Standard pairs are combinatorial structures that contain information on primary and irreducible decompositions of monomial ideals, as well as multiplicities. One of the main contributions of [Matusevich and Yu 2020] is that standard pairs and the associated algebraic concepts can be effectively computed over affine semigroup rings.

The `SageMath` library `StdPairs` implements the algorithms of [Matusevich and Yu 2020] to calculate standard pairs for monomial ideals in any pointed (nonnormal) affine semigroup ring. This library can be regarded as a generalization of the `standardPairs` function in `Macaulay2` implemented by [Hoşten and Smith 2002]. This library is provided as an online supplement to this paper.

Outline. Section 2 provides background on affine semigroup rings, their monomial ideals, and related combinatorial notions. It also explains their implementation as `SageMath` classes. Section 3 presents the implementation of algorithms to find standard pairs, proposed in [Matusevich and Yu 2020, Section 4]. Section 4 shows compatibility with the `Normaliz` package by introducing methods to translate objects in `SageMath` into objects in `Macaulay2` using `Normaliz`.

MSC2020: primary 13F65, 68W30, 90C90; secondary 13P25, 20M25.
Keywords: affine semigroups, standard pairs, monomial ideals, semigroup rings.
`StdPairs` version 1.0

1.1. ***Notation.*** We denote a semigroup of nonnegative integers including 0 by $\mathbb{N} = \{0, 1, 2, \ldots\}$. A ring of integers is $\mathbb{Z}$. All nonnegative real numbers are represented by $\mathbb{R}_{\geq 0}$. Boldface uppercase letters and boldface lowercase letters denote matrices and vectors, respectively. This will be used when we call a cone $\mathbb{R}_{\geq 0}\boldsymbol{A}$ for some $d \times n$ matrix $\boldsymbol{A}$ over $\mathbb{Z}$. Let $\mathbb{K}$ be an arbitrary field.

2. AFFINE SEMIGROUP, IDEAL, AND PROPER PAIR AS CLASSES OF SAGEMATH.

2.1. ***Mathematical background.*** An *affine semigroup* is a semigroup of $\mathbb{Z}^d$ generated by finitely many vectors $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ for some $n \in \mathbb{N}$. We let $\boldsymbol{A}$ be a $d \times n$ matrix whose column vectors are $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{Z}^d$. The set of all nonnegative integer linear combinations of $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$, denoted by $\mathbb{N}\boldsymbol{A}$, is an affine semigroup. These columns $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ are the *generators* of the affine semigroup $\mathbb{N}\boldsymbol{A}$; the matrix $\boldsymbol{A}$ is called the *generating matrix*. Since $\mathbb{N}\boldsymbol{A}$ contains 0, an affine semigroup is a commutative monoid. Given a field $\mathbb{K}$, we are concerned with the *affine semigroup ring* $\mathbb{K}[\mathbb{N}\boldsymbol{A}]$. A natural first example is the polynomial ring in d-variables; in this case, $\boldsymbol{A}$ is the $d \times d$ identity matrix. We refer to [Miller and Sturmfels 2005, Section 7] for more background on this topic. Throughout this article, we assume that the affine semigroup $\mathbb{N}A$ under consideration is *pointed*, which means that the cone $\mathbb{R}_{\geq 0}\boldsymbol{A}$ does not contain lines.

An *ideal* of an affine semigroup is a set $I \subset \mathbb{N}\boldsymbol{A}$ such that $I + \mathbb{N}\boldsymbol{A} \subseteq I$. There is a one-to-one correspondence between monomial ideals of $\mathbb{K}[\mathbb{N}\boldsymbol{A}]$ and ideals of $\mathbb{N}\boldsymbol{A}$. Therefore, the definition of prime, irreducible, and primary ideals of $\mathbb{K}[\mathbb{N}\boldsymbol{A}]$ can be naturally extended to the ideals of an affine semigroup. The *standard monomials* of an ideal $I \subset \mathbb{N}\boldsymbol{A}$ are all elements of $\mathbb{N}\boldsymbol{A} \setminus I$. Let $\mathrm{std}(I)$ be a set of all standard monomials with respect to I.

A *face* of an affine semigroup $\mathbb{N}\boldsymbol{A}$ is a subsemigroup $\mathbb{N}\boldsymbol{F} \subseteq \mathbb{N}\boldsymbol{A}$ such that the complement $\mathbb{N}\boldsymbol{A} \setminus \mathbb{N}\boldsymbol{F}$ is an ideal of $\mathbb{N}\boldsymbol{A}$ [Miller and Sturmfels 2005, Definition 7.8]. Equivalently, it is a subsemigroup $\mathbb{N}\boldsymbol{F}$ with the property that $\boldsymbol{a} + \boldsymbol{b} \in \mathbb{N}\boldsymbol{F}$ if and only if $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}\boldsymbol{F}$. The faces of an affine semigroup form a lattice which is isomorphic to the face lattice of a (real) cone over the affine semigroup [Bruns and Herzog 1993; Miller and Sturmfels 2005]. Thus, we may represent a face $\mathbb{N}\boldsymbol{F}$ as a submatrix $\boldsymbol{F}$ of $\boldsymbol{A}$.

A *pair* is a tuple $(\boldsymbol{a}, \boldsymbol{F})$ of an element $\boldsymbol{a}$ in $\mathbb{N}\boldsymbol{A}$ and a face $\boldsymbol{F}$ of $\mathbb{N}\boldsymbol{A}$ [Matusevich and Yu 2020]. A *proper pair* of an ideal I is a pair $(\boldsymbol{a}, \boldsymbol{F})$ such that $\boldsymbol{a} + \mathbb{N}\boldsymbol{F} \subseteq \mathrm{std}(I)$. A pair $(\boldsymbol{a}, \boldsymbol{F})$ *divides* $(\boldsymbol{b}, \boldsymbol{G})$ if there exists $\boldsymbol{c} \in \mathbb{N}\boldsymbol{A}$ such that $\boldsymbol{a} + \boldsymbol{c} + \mathbb{N}\boldsymbol{F} \subseteq \boldsymbol{b} + \mathbb{N}\boldsymbol{G}$ [Matusevich and Yu 2020]. The set of all proper pairs of an ideal I is partially ordered $\prec$ by inclusion. In other words, $(\boldsymbol{a}, \boldsymbol{F}) \prec (\boldsymbol{b}, \boldsymbol{G})$ if $\boldsymbol{a} + \mathbb{N}\boldsymbol{F} \subset \boldsymbol{b} + \mathbb{N}\boldsymbol{G}$. The *standard pairs* of an ideal I are the maximal elements of the set of all proper pairs of I with this partial order. We denote by $\mathrm{Std}(I)$ the set of all standard pairs of an ideal I.

We remark that our notation here differs from existing notation for standard pairs over polynomial rings. Over the polynomial ring $\mathbb{K}[x_1, x_2, \ldots, x_n]$, a pair is a tuple $(x^{\boldsymbol{a}}, V)$ where $x^{\boldsymbol{a}}$ is a monomial $x_1^{a_1} x_2^{a_2} \cdots x_n^{a_n}$ for some integer vector $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ and V is a set of variables [Hoşten and Smith 2002; Sturmfels et al. 1995]. From the viewpoint of affine semigroup rings, the polynomial ring is a special case when the underlying affine semigroup is generated by an $n \times n$ identity matrix $\boldsymbol{I}$. Since the

cone $\mathbb{R}_{\geq 0}\boldsymbol{I}$ is a simplicial cone, i.e., every subset of rays form a face, we may interpret V as a face. The following example shows the different notations for the standard pairs of a monomial ideal

$$I = \left\langle x^{\begin{bmatrix}1\\3\\1\end{bmatrix}}, x^{\begin{bmatrix}1\\2\\2\end{bmatrix}}, x^{\begin{bmatrix}0\\3\\2\end{bmatrix}}, x^{\begin{bmatrix}0\\2\\3\end{bmatrix}} \right\rangle$$

in the polynomial ring $\mathbb{K}[x_1, x_2, x_3]$:

In `Macaulay2`,

```
i1 : R = QQ[x,y,z];
i2 : I = monomialIdeal(x*y^3*z, x*y^2*z^2, y^3*z^2, y^2*z^3)
                        3      2 2   3 2   2 3
o2 = monomialIdeal (x*y z, x*y z , y z , y z )
o2 : MonomialIdeal of R
i3 : standardPairs I
o3 = {{1, {x, z}}, {y, {x, z}}, {1, {x, y}}, {z, {y}},
  2          2 2
{y z, {x}}, {y z , {}}}
o3 : List
```

whereas in the given library `StdPairs` in `SageMath`,

```
sage: from stdpairs import *
sage: A = matrix(ZZ,[[1,0,0],[0,1,0],[0,0,1]])
sage: Q = AffineMonoid(A)
sage: M = matrix(ZZ,[[1,1,0,0],[3,2,3,2],[1,2,2,3]])
sage: I = MonomialIdeal(M,Q)
sage: I.standard_cover()
{(1,): [([[0], [0], [1]]^T,[[0], [1], [0]])],
 (0, 2): [([[0], [1], [0]]^T,[[1, 0], [0, 0], [0, 1]]),
  ([[0], [0], [0]]^T,[[1, 0], [0, 0], [0, 1]])],
 (0, 1): [([[0], [0], [0]]^T,[[1, 0], [0, 1], [0, 0]])],
 (): [([[0], [2], [2]]^T,[[], [], []])],
 (0,): [([[0], [2], [1]]^T,[[1], [0], [0]])]}
```

`()`, `(0,)`, `(0, 2)`, `(0, 1)`, and `(1,)` in `StdPairs` of `SageMath` are indices of columns of `A`. These denote `{}`, `{x}`, `{x, y}`, `{x, z}`, and `{y}`, respectively, in `Macaulay2`. Therefore, for example, the pair `([[0], [0], [1]]^T,[[0], [1], [0]])` will represent `{z, {y}}`, while the pair `([[0], [0], [0]]^T,[[1, 0], [0, 0], [0, 1]])` represents `{1, {x, z }}`, and so on. Therefore, this example shows that `StdPairs` is consistent with `Macaulay2`.

2.2. ***Classes in StdPairs.*** We implement three classes related to affine semigroups, semigroup ideals, and proper pairs respectively. This implementation is based on `SageMath` 9.1 with `Python` 3.7.3. and the `4ti2` package. Detailed usage and examples of each method or object can be found using the command `<method_name>?` in `SageMath` or https://byeongsuyu.github.io/StdPairs/, the documentation of `StdPairs` made by the `Sphinx` package.

Class AffineMonoid. This class is constructed by using an integer matrix $\boldsymbol{A}$. The name follows the convention of `SageMath` which distinguishes monoid from semigroup. In `SageMath`, $\boldsymbol{A}$ can be expressed

as a 2-dimensional `NumPy.ndarray` type or an integer matrix of `SageMath`. For example,

```
sage: from stdpairs import *
sage: A = matrix(ZZ,[[1,2],[0,2]])
sage: Q = AffineMonoid(A)
```

generates Q as a type of `AffineMonoid`. This class has several methods as explained below:

- `Q.gens()` returns a matrix generating an affine monoid Q as `NumPy.ndarray` type. This may not be a minimal generating set of Q.
- `Q.mingens()` returns a minimal generating matrix of an affine monoid of Q.
- `Q.poly()` returns a real cone $\mathbb{R}_{\geq 0}Q$ represented as a type of `Polyhedron` in `SageMath`. If one generates Q with `True` parameter, i.e.,

  ```
  sage: from stdpairs import *
  sage: A = matrix(ZZ,[[1,2],[0,2]])
  sage: Q = AffineMonoid(A,is_normaliz=True)
  ```

  then `Q.poly()` is of a class of `Normaliz` integral polyhedron. This requires the `PyNormaliz` package. See [Köppe and Labbé 2019] for more details.
- `Q.face_lattice()` returns a finite lattice containing all faces of the affine semigroup. A face in the lattice is saved as a tuple storing column numbers of generators $\boldsymbol{A}$. This lattice is of type `FiniteLatticePoset` in `SageMath`. For example,

  ```
  sage: Q.face_lattice()
  Finite lattice containing 5 elements
  sage: Q.face_lattice().list()
  [(-1,), (), (0,), (1,), (0, 1)]
  ```
- `Q.index_to_face()` returns a `dictionary` type object whose keys are tuples denoting indices of column vectors consisting of faces, and whose items are corresponding faces of `Q.poly()`. For example,

  ```
  sage: Q.index_to_face()
  {(-1,): A -1-dimensional face of a Polyhedron in ZZ^2,
   (): A 0-dimensional face of a Polyhedron in ZZ^2
   defined as the convex hull of 1 vertex,
   (0,): A 1-dimensional face of a Polyhedron in ZZ^2
   defined as the convex hull of 1 vertex and 1 ray,
   (1,): A 1-dimensional face of a Polyhedron in ZZ^2
   defined as the convex hull of 1 vertex and 1 ray,
   (0,1): A 2-dimensional face of a Polyhedron in ZZ^2
    defined as the convex hull of 1 vertex and 2 rays}
  ```
- `Q.index_of_face(matrix face)` returns a face as a tuple of indices of column vectors of a generator $\boldsymbol{A}$ corresponding to a given submatrix `face` of $\boldsymbol{A}$. For example,

  ```
  sage: M = matrix(ZZ,[[2],[2]])
  sage: Q.index_of_face(M)
  (1,)
  ```

  Here, `face` should be a submatrix of `Q.gens()` which form a face.

- `Q.face(tuple index)` returns a face as a submatrix of a generator $\boldsymbol{A}$ corresponding to a given tuple `index`. For example,

```
sage: Q.face((1,))
array([[2],
       [2]])
```

- `Q.integral_support_vectors()` return a `dictionary` type object whose keys are tuples denoting faces and whose items are integral support functions of facets containing $\boldsymbol{F}$ as a vector form. An *integral support function* $\phi_{\boldsymbol{H}}$ of a facet $\boldsymbol{H}$ is a linear function $\phi_{\boldsymbol{H}} : \mathbb{R}^d \to \mathbb{R}$ such that $\phi_{\boldsymbol{H}}(\mathbb{Z}^d) = \mathbb{Z}$, $\phi_{\boldsymbol{H}}(\boldsymbol{a}) \geq 0$ for all column vectors $\boldsymbol{a}$ of generators A, and $\phi_{\boldsymbol{H}}(\boldsymbol{a}) = 0$ if and only if $\boldsymbol{a} \in \boldsymbol{H}$. By linearity, $\phi_{\boldsymbol{H}}(\boldsymbol{a}) = \boldsymbol{b} \cdot \boldsymbol{a}$ for some rational vector $\boldsymbol{b}$. We call $\boldsymbol{b}$ an *integral support vector*. Each item of `Q.integral_support_vectors()` is a matrix as `NumPy.ndarray` type whose rows are integral support vectors of facets containing the given face. For example,

```
sage: Q.integral_support_vectors()
{(): array([[ 0,  1],
        [ 1, -1]]),
 (0,): array([[0, 1]]),
 (1,): array([[ 1, -1]]),
 (0, 1): array([], dtype=int64)}
```

  In this code, `()` denoting 0 has two integral support vectors, since it is an intersection of two facets `(0,)` and `(1,)`, while `(0,1)` has no such integral support vectors since it is not a proper face but the affine semigroup itself. See [Matusevich and Yu 2020, Definition 2.1] for the precise definition of a (primitive) integral support function.

- `Q.is_empty()` returns a boolean value indicating whether Q is a trivial affine semigroup or not. A *trivial affine semigroup* is an empty set as an affine semigroup.

- `Q.is_pointed()` returns a boolean value indicating whether Q is a pointed affine semigroup or not.

- `Q.is_element(vector b)` returns nonnegative integral inhomogeneous solutions (minimal integer solutions) of $\boldsymbol{Ax} = \boldsymbol{b}$ using `zsolve` in [4ti2]. If $\boldsymbol{b}$ is not an element of an affine semigroup Q, then it returns an empty matrix. The vector $\boldsymbol{b}$ should be a `NumPy.ndarray` type 2-dimensional object with one column, or a matrix of `SageMath` with only one column.

- `Q.save_txt()` returns a string containing information about `Q`. This can be loaded again using `txt_to_affinemonoid(string info)`, which will be explained in Section 2.3.

- `Q.save(string path)` saves the given object `Q` as binary file on the given path. This can be loaded again using `load(path)`, a pre-existing global function of `SageMath`.

Moreover, one can directly compare affine semigroups using the equality operator `==` in `SageMath`.

Class MonomialIdeal. This class is constructed by an affine semigroup Q and generators of an ideal as a matrix form, say $\boldsymbol{M}$, which is a 2-dimensional `NumPy.ndarray` object or an integer matrix of `SageMath`.

For example,

```
sage: M = matrix(ZZ,[[4,6],[4,6]])
sage: I = MonomialIdeal(M,Q)
sage: I
An ideal whose generating set is
[[4]
 [4]]
```

As shown in the example above, this class stores only minimal generators of the ideal. The attributes and methods are explained below:

- `I.gens()` returns the minimal generators of I as a `NumPy.ndarray` type object.
- `I.ambient_monoid()` returns the ambient affine semigroup of I.
- `I.standard_cover(verbose = False)` returns the *standard cover* of I, a `dictionary` object whose keys are faces and whose items are lists consisting of `ProperPair` type objects whose face is equal to the corresponding key. `ProperPair` objects will be explained in Section 2.2. The definition of the standard cover will be given in Section 3.1. Users can check the progress of the computation if `verbose=True`.
- `I.overlap_classes()` returns a `dictionary` object whose keys are tuples denoting faces and whose items are lists of lists representing overlap classes of I. An *overlap class* of an ideal I is a set of standard pairs such that their representing submonoids intersect nontrivially.
- `I.maximal_overlap_classes()` returns all maximal overlap classes of I with divisibility. An overlap class is *maximal with divisibility* if every pair in the overlap class can divide only pairs in itself. See [Matusevich and Yu 2020, Section 3] for the detail.
- `I.irreducible_decomposition()` returns a list of components of the irredundant irreducible primary decomposition of I.
- `I.associated_primes()` returns all associated prime ideals of I as a `dictionary` type. In other words, the function returns a dictionary whose keys are faces of the affine semigroup as a `tuple` and whose values are associated prime ideals corresponding to the face in its key.
- `I.multiplicity(ideal P or face F)` returns a multiplicity of I over the given associated prime P. Since there is a one-to-one correspondence between monomial prime ideals and faces of an affine semigroup, this method takes the face $\boldsymbol{F}$ (as a tuple) corresponding to a prime ideal P as a valid input instead.
- `I.is_element(vector b)` returns nonnegative integral inhomogeneous solutions (minimal integer solutions) of $\boldsymbol{Ax} = \boldsymbol{b} - \boldsymbol{a}$ for each generator $\boldsymbol{a}$ of I using `zsolve` in [4ti2]. If $\boldsymbol{b}$ is an element of the ideal, then it returns a list $[\boldsymbol{x}, \boldsymbol{a}]$ for some generator $\boldsymbol{a}$ such that $\boldsymbol{a} + \boldsymbol{Ax}^T = \boldsymbol{b}$. Otherwise, it returns an empty matrix. The vector $\boldsymbol{b}$ should be a `NumPy.ndarray` type 2-dimensional object with one column, or a matrix of `SageMath` with only one column.

- `I.is_standard_monomial(vector b)` returns a boolean value indicating whether the given vector $\boldsymbol{b}$ is a standard monomial or not.
- `I.is_principal()` returns a boolean value indicating whether I is principal or not. Likewise, the similar methods `I.is_empty()`, `I.is_irreducible()`, `I.is_primary()`, `I.is_prime()`, and `I.is_radical()` return a boolean value indicating whether I has the properties implied by their name or not.
- `I.radical()` returns the radical of I as a `MonomialIdeal` object.
- `I.intersect(J)` returns an intersection of two ideals I and J as a `MonomialIdeal` object. Likewise, addition `+`, multiplication $*$, and comparison `==` are defined between two objects. The following example shows an addition of two monomial ideals in `SageMath`:

```
sage: I = MonomialIdeal(matrix(ZZ,[[4,6],[4,6]]),Q)
sage: J = MonomialIdeal(matrix(ZZ,[[5],[0]]),Q)
sage: I.intersect(J)
An ideal whose generating set is
[[9]
 [4]]
sage: I+J
An ideal whose generating set is
[[5 4]
 [0 4]]
 sage: I*J
An ideal whose generating set is
[[9]
 [4]]
```

- `I.save_txt()` returns a string which can be used to recover the object I and its precalculated properties without calculation. That is, it contains not only the generators of I, but also its standard cover, overlap classes, associated primes, and irreducible primary decompositions if they were calculated. This can be loaded again using `txt_to_monomialideal(string info)` (see Section 2.3).
- `I.save(string path)` saves the given object `I` as a binary file on the given path. This can be loaded again using `load(path)`, a pre-existing global function of `SageMath`.

Class ProperPair. A proper pair $(\boldsymbol{a}, \boldsymbol{F})$ of an ideal I can be declared in `SageMath` by specifying an ideal I, a standard monomial $\boldsymbol{a}$ as a matrix form (or `NumPy` 2D array), and a face $\boldsymbol{F}$ as a tuple. If $(\boldsymbol{a}, \boldsymbol{F})$ is not proper, then `SageMath` calls a `ValueError`. The following example shows two ways of defining a proper pair:

```
sage: import numpy as np
sage: I = MonomialIdeal(matrix(ZZ,[[4,6],[4,6]]),Q)
sage: PP = ProperPair(np.array([2,0])[np.newaxis].T,(0,),I)
sage: PP
([[2], [0]]^T,[[1], [0]])
sage: QQ = ProperPair(np.array([2,0])[np.newaxis].T,(0,),I,
....: properness =True)
sage: QQ
([[2], [0]]^T,[[1], [0]])
```

The second line tests whether the pair is a proper pair of the given ideal I or not before generating PP. However, the fourth line with properness=True generates QQ without checking whether QQ is a proper pair of I or not. Use the third parameter with True only if the generating pair is proper a priori. In any case, each PP and QQ denotes a proper pair whose initial monomial is $\left[\begin{smallmatrix}2\\0\end{smallmatrix}\right]$ and whose face is $\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$.

The attributes and methods are explained below. We assume that PP denotes a proper pair $(\boldsymbol{a}, \boldsymbol{F})$.

- PP.monomial(), PP.face(), and PP.ambient_ideal() return the initial monomial $\boldsymbol{a}$ (as a NumPy 2D array), the face $\boldsymbol{F}$ (as a tuple), and its ambient ideal (as an object of AffineMonoid) respectively.
- PP.is_maximal() returns a boolean value indicating whether the given pair is maximal with respect to the divisibility of proper pairs of the ambient ideal. If PP is generated without testing whether its monomial is in the given ideal I or not, this method raises a warning instead of returning a boolean value.
- PP.is_element(vector $\boldsymbol{b}$) returns nonnegative integral inhomogeneous solutions (minimal integer solutions) of $\boldsymbol{a} + \boldsymbol{F}\boldsymbol{x} = \boldsymbol{b}$ using zsolve in [4ti2]. If $\boldsymbol{b}$ is not an element of the submonoid $\boldsymbol{a} + \mathbb{N}\boldsymbol{F}$, then it returns an empty matrix.
- Like AffineMonoid or MonomialIdeal, one can directly compare proper pairs using the equality operator == in SageMath.

2.3. *Global functions.*

- prime_ideal(tuple face, AffineMonoid Q) returns a prime ideal of the given AffineMonoid object Q corresponding to the tuple object face as an object of MonomialIdeal.

```
sage: prime_ideal((1,),Q)
An ideal whose generating set is
[[1]
 [0]]
```

- div_pairs(pair PP, pair QQ) will return a matrix whose column $\boldsymbol{u}$ is a minimal solution of $\boldsymbol{a} + \boldsymbol{u} + \mathbb{N}\boldsymbol{F} \subseteq \boldsymbol{b} + \mathbb{N}\boldsymbol{G}$ if $PP = (\boldsymbol{a}, \boldsymbol{F})$ and $QQ = (\boldsymbol{b}, \boldsymbol{G})$. The returned value is a nonempty matrix if and only if a pair PP divides a pair QQ. For example, suppose two pairs PP and QQ are $\left[\begin{smallmatrix}2\\0\end{smallmatrix}\right] + \mathbb{N}\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}3\\0\end{smallmatrix}\right] + \mathbb{N}\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]$ respectively. Then,

```
sage: I = MonomialIdeal(matrix(ZZ,[[4,6],[4,6]]),Q)
sage: PP = ProperPair(matrix(ZZ,[[2],[0]]),(0,),I)
sage: QQ = ProperPair(matrix(ZZ,[[3],[0]]),(0,),I)
sage: div_pairs(PP,QQ)
[1]
[0]
sage: div_pairs(QQ,PP)
[0]
[0]
```

since $\left(\left[\begin{smallmatrix}2\\0\end{smallmatrix}\right] + \mathbb{N}\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\right) \supseteq \left(\left[\begin{smallmatrix}3\\0\end{smallmatrix}\right] + \mathbb{N}\left[\begin{smallmatrix}1\\0\end{smallmatrix}\right]\right)$.

- `txt_to_affinemonoid(string info)` (or `txt_to_monomialideal(string info)`) loads an `AffineMonoid` object (or a `MonomialIdeal` object) in the string `info`, which is generated by the `AffineMonoid.save_txt()` (or `MonomialIdeal.save_txt()`) methods. These are useful for users who want to avoid repeating calculations which were previously done. For example, the ideal J in

```
sage: I = MonomialIdeal(matrix(ZZ,[[4,2],[4,0]]),Q)
sage: I.standard_cover()
{(): [([[1], [0]]^T,[[], []]),
  ([[0], [0]]^T,[[], []]),
  ([[3], [2]]^T,[[], []]),
  ([[2], [2]]^T,[[], []])]}
sage: J=txt_to_monomialideal(I.save_txt())
sage: J.standard_cover()
{(): [([[1], [0]]^T,[[], []]),
  ([[0], [0]]^T,[[], []]),
  ([[3], [2]]^T,[[], []]),
  ([[2], [2]]^T,[[], []])]}
```

  is a new `MonomialIdeal` object; however, it does not need time to calculate its standard cover, since precalculated information of the standard cover was stored in `I.save_txt()` and transferred to J.

- `pair_difference(ProperPair PP, ProperPair QQ)` is a global function which decomposes $PP \setminus QQ$ as a finite union of pairs. See Theorem 1 and subsequent arguments for details.

- `from_macaulay2(string var_name)` and `to_macaulay2(MonomialIdeal I)` are global functions used for communicating with `Macaulay2` objects. See Section 4 for details.

3. Implementation of an algorithm finding standard pairs.

3.1. ***Case 1: Principal ideal.*** A *cover* of standard monomials of an ideal I is a set of proper pairs of I such that the union of all subsemigroups $\boldsymbol{a}+\mathbb{N}\boldsymbol{F}$ corresponding to an element $(\boldsymbol{a}, \boldsymbol{F})$ of the cover is equal to the set of all standard monomials. The *standard cover* of an ideal I is a cover of I whose elements are standard pairs. The standard cover of a monomial ideal I is unique by the maximality of standard pairs among all proper pairs of I. A key idea in [Matusevich and Yu 2020, Section 4] is to construct covers containing all standard pairs. Once a cover is obtained, we can then produce the standard cover.

The following result helps to compute the standard cover in the special case of a principal ideal.

Theorem 1 [Matusevich and Yu 2020, Theorem 4.1]. *Let $\boldsymbol{b}, \boldsymbol{b}' \in \mathbb{N}\boldsymbol{A}$ and let $\boldsymbol{G}, \boldsymbol{G}'$ be faces of A such that $\boldsymbol{G} \cap \boldsymbol{G}' = \boldsymbol{G}$. There exists an algorithm to compute a finite collection C of pairs over faces of $\boldsymbol{G}$ such that*

$$(\boldsymbol{b}+\boldsymbol{G}) \setminus (\boldsymbol{b}'+\boldsymbol{G}') = \bigcup_{(\boldsymbol{a},\boldsymbol{F})\in C} (\boldsymbol{a}+\boldsymbol{F}).$$

The *pair difference* of the pairs $(\boldsymbol{b}, \boldsymbol{G})$ and $(\boldsymbol{b}', \boldsymbol{G}')$ is a finite collection of pairs over faces of $\boldsymbol{G}$ given by Theorem 1.

Corollary 2. *Given a principal ideal $I = \langle \boldsymbol{b} \rangle$, the pair difference of pairs $(0, \boldsymbol{A})$ and $(\boldsymbol{b}, \boldsymbol{A})$ is the standard cover of I.*

Proof. Theorem 1 implies that the pair difference is a cover of I. To see it is the standard cover, suppose that the ambient affine semigroup is generated by $\boldsymbol{A} = [\boldsymbol{a}_1 \cdots \boldsymbol{a}_n]$. Let $(\boldsymbol{c}, F)$ be a proper pair in the pair difference. Without loss of generality, we assume that $F = [\boldsymbol{a}_1 \cdots \boldsymbol{a}_m]$ for some $m < n$ by renumbering indices. By the proof of Theorem 1 in [Matusevich and Yu 2020], $\boldsymbol{c} = A \cdot \boldsymbol{u}$ where $x^{\boldsymbol{u}} \in \mathbb{K}[\mathbb{N}^n]$ is a standard monomial such that $(x^{\boldsymbol{u}}, \{x_1, \ldots, x_m\})$ is a standard pair of some monomial ideal J in $\mathbb{K}[\mathbb{N}^n]$.

Suppose that there exists $(\boldsymbol{d}, \boldsymbol{G})$ such that $F \subseteq \boldsymbol{G}$ and $\boldsymbol{d} + \boldsymbol{g} = \boldsymbol{c}$ for some $\boldsymbol{g} \in G$. Since $\boldsymbol{d} \in \mathbb{N}\boldsymbol{A}$, $\boldsymbol{d} = A\boldsymbol{w}$ for some $\boldsymbol{w} \in \mathbb{N}^n$. Since A is pointed, $\boldsymbol{w}$ is, coordinatewise, less than $\boldsymbol{u}$. Thus, $(x^{\boldsymbol{w}}, \{x_1, \ldots, x_m\})$ contains $(x^{\boldsymbol{u}}, \{x_1, \ldots, x_m\})$. Lastly, $(x^{\boldsymbol{w}}, \{x_1, \ldots, x_m\})$ is a proper pair of J, otherwise, there exists $x^{\boldsymbol{v}} \in \mathbb{K}[x_1, \ldots, x_m] \subseteq \mathbb{K}[\mathbb{N}^n]$ such that $x^{\boldsymbol{w}+\boldsymbol{v}} \in J$. Then, $x^{\boldsymbol{g}} x^{\boldsymbol{w}+\boldsymbol{v}} \in J \Longrightarrow x^{\boldsymbol{u}+\boldsymbol{v}} \in J \cap (x^{\boldsymbol{u}}, \{x_1, \ldots, x_m\}) = \varnothing$ leads to a contradiction.

Thus, by maximality of the standard pair, $\boldsymbol{w} = \boldsymbol{u}$. This implies $\boldsymbol{d} = \boldsymbol{c}$. Moreover, $G = F$, otherwise there exists $j \in \{1, 2, \ldots, n\} \setminus \{1, \ldots, m\}$ such that $x^{\boldsymbol{u}} x_j^l \notin J$ for any l, which implies that $(x^{\boldsymbol{u}}, \{x_1, \ldots, x_m, x_j\})$ is a proper pair of J strictly containing a standard pair $(x^{\boldsymbol{u}}, \{x_1, \ldots, x_m\})$ of J, a contradiction. □

Theorem 1 is implemented as a method `pair_difference((`$\boldsymbol{b}$`,` $\boldsymbol{F}$`), (`$\boldsymbol{b}'$`,` $\boldsymbol{F}'$`))` within the library `StdPairs`. The two input arguments should be of type `ProperPair`. It returns the pair difference of pairs $(\boldsymbol{b}, \boldsymbol{F})$ and $(\boldsymbol{b}', \boldsymbol{F}')$ with `dictionary` type, called `Cover`. `Cover` classifies pairs by their faces. For example, the code below shows the pair difference of pairs $(0, \boldsymbol{A})$ and $((0, 2), \boldsymbol{A})$, which are

$$\left(0, \begin{bmatrix}2\\0\end{bmatrix}\right), \left(\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}\right), \left(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}\right), \text{ and } \left(\begin{bmatrix}1\\1\end{bmatrix}, \begin{bmatrix}2\\0\end{bmatrix}\right).$$

```
sage: from stdpairs import *
sage: Q = AffineMonoid(matrix(ZZ, [[2,0,1],[0,1,1]]))
sage: I = MonomialIdeal(matrix(ZZ,0),Q)
sage: C = ProperPair(np.array([[0,0]]).T, (0,1,2), I )
sage: D = ProperPair(np.array([[0,2]]).T, (0,1,2), I )
sage: print(pair_difference(C,D))
{(0,): [([[1], [2]]^T,[[2], [0]]), ([[1], [1]]^T,[[2], [0]]),
([[0], [1]]^T,[[2], [0]]), ([[0], [0]]^T,[[2], [0]])]}
```

By Corollary 2, this is the standard cover of the ideal $I = \langle (0, 2) \rangle$ in an affine semigroup $\mathbb{N}\left[\begin{smallmatrix}2&0&1\\0&1&1\end{smallmatrix}\right]$.

The method `pair_difference((`$\boldsymbol{b}$`,` $\boldsymbol{F}$`), (`$\boldsymbol{b}'$`,` $\boldsymbol{F}'$`))` uses `standardPairs` of `Macaulay2` internally to find standard pairs of a polynomial ring, which is implemented by [Hoşten and Smith 2002]. Briefly, the method `pair_difference((`$\boldsymbol{b}$`,` $\boldsymbol{F}$`), (`$\boldsymbol{b}'$`,` $\boldsymbol{F}'$`))` calculates minimal solutions of the integer linear system

$$\begin{bmatrix} F & -F' \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{v} \end{bmatrix} = \boldsymbol{b}' - \boldsymbol{b}$$

using `zsolve` in `4ti2`. The solutions form an ideal J of a polynomial ring in the proof of Theorem 1 on `Macaulay2`. Next, `standardPairs` derives standard pairs of J. Lastly, the method `pair_difference`

constructs proper pairs based on the standard pairs of J, and classifies the proper pairs based on their faces and returns the pair difference.

3.2. ***Case 2: General ideal.*** [Matusevich and Yu 2020, Proposition 4.4] gives an algorithm to find the standard cover of nonprincipal monomial ideals.

Proposition 3 [Matusevich and Yu 2020, Proposition 4.4]. *Let I be a monomial ideal in $\mathbb{K}[\mathbb{N}\boldsymbol{A}]$. There is an algorithm whose input is a cover of the standard monomials of I, and whose output is the standard cover of I.*

According to the proof of [Matusevich and Yu 2020, Proposition 4.4], this is achieved by repeating the procedures below.

(1) Input: C_0, an initial cover of I.

(2) For each $(\boldsymbol{a}, F) \in C_0$, find minimal solutions of $(\boldsymbol{a} + \mathbb{R}F) \cap \mathbb{N}\boldsymbol{A}$ using the primitive integral support functions. (See [Matusevich and Yu 2020, Lemma 4.2] for the detail.)

If $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_m$ are the minimal solutions of $(\boldsymbol{a} + \mathbb{R}F) \cap \mathbb{N}\boldsymbol{A}$, then we construct pairs such as $(\boldsymbol{b}_1, F), (\boldsymbol{b}_2, F), \ldots, (\boldsymbol{b}_m, F)$ and store them in the attribute C_1.

(3) For each pair $(\boldsymbol{b}, F) \in C_1$, construct $(\boldsymbol{b}, G)$ for any face G which is not strictly contained in F. If $(\boldsymbol{b}, G)$ is a proper pair of I, save $(\boldsymbol{b}, G)$ on the attribute C_2.

(4) If C_0 is equal to C_2, we are done. Otherwise, set $C_0 := C_2$ and repeat the above process.

The method `_czero_to_cone(`C_0, I`)` in the hidden module `_stdpairs` of `StdPairs` implements (2) to return C_1. It calls the method `_minimal_holes(vector` $\boldsymbol{a}$`, face F, affine semigroup A)` internally, which is the implementation of Lemma 4.2 in [Matusevich and Yu 2020]. The method `_cone_to_ctwo(`C_1, I`)` implements (3). Since the constructor function of the class `ProperPair` checks whether the pair is proper or not, the method `_cone_to_ctwo(`C_1, I`)` tries to construct proper pairs as an attribute in `SageMath` and records them if it is successful.

Now we are ready to find the standard cover of a general ideal I whose minimal generators are $\langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \rangle$. One can find standard pairs as in [Matusevich and Yu 2020, Theorem 4.5] as described below:

(1) Find the standard cover C of $\langle \boldsymbol{b}_1 \rangle$ using pair difference.

(2) For $i = 2$ to n:

  (a) For each pair $(\boldsymbol{b}, F)$ in C, replace it with elements of the pair difference of pairs $(\boldsymbol{b}, F)$ and $(\boldsymbol{b}_i, A)$. After this process, C is a cover of an ideal $\langle \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_i \rangle$.

  (b) Using the algorithm of Proposition 3, find the standard cover C' of $\langle \boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_i \rangle$.

  (c) Replace C with C'.

(3) Return C.

The returned value C is now the standard cover of I.

The library `StdPairs` implements [Matusevich and Yu 2020, Theorem 4.5] as a hidden method `_standard_pairs(`I`)`. This method has an input I whose type is `MonomialIdeal`. It returns a cover whose type is `dictionary`, classifying standard pairs by its face. For example, the code below shows that the standard cover of an ideal generated by

$$\begin{bmatrix} 2 & 2 & 2 \\ 0 & 1 & 2 \\ 2 & 2 & 2 \end{bmatrix}$$

in an affine semigroup

$$\mathbb{N}\boldsymbol{A} = \mathbb{N}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

is

$$\left\{\left(0, \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}\right), \left(\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}\right), \quad \text{and} \quad \left(\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}\right)\right\}.$$

```
sage: from stdpairs import *
sage: Q=AffineMonoid(matrix(ZZ,[[0,1,1,0],[0,0,1,1],[1,1,1,1]]))
sage: I=MonomialIdeal(matrix(ZZ,[[2,2,2],[0,1,2],[2,2,2]]),Q)
sage: I.standard_cover()
{(0, 3): [([[1], [0], [1]]^T,[[0, 0], [0, 1], [1, 1]]),
  ([[1], [1], [1]]^T,[[0, 0], [0, 1], [1, 1]]),
  ([[0], [0], [0]]^T,[[0, 0], [0, 1], [1, 1]])]}
```

4. COMPATIBILITY WITH NORMALIZ PACKAGE IN SAGEMATH AND MACAUALAY2. `Normaliz` is a package in `SageMath` and `Macaulay2` for finding Hilbert bases of rational cones and their normal affine monoid [Bruns and Ichim 2010]. `StdPairs` has methods translating classes in Section 2 into objects in the `Normaliz` package. If an affine semigroup $\mathbb{N}\boldsymbol{A}$ is *normal*, i.e., $\mathbb{N}\boldsymbol{A} = \mathbb{Z}^d \cap \mathbb{R}_{\geq 0}A$, then this translation works well. However, if it is not normal, then this translates $\mathbb{N}\boldsymbol{A}$ into its saturation described in Section 2.

For `SageMath`, one can have a polyhedron over $\mathbb{Z}$ with the `Normaliz` package in `SageMath` by adding an argument `True` on the constructor of `AffineMonoid`. For example, the code below gives an `AffineMonoid` class attribute Q whose attribute `Q.poly()` is a polyhedron over $\mathbb{Z}$ with `Normaliz`. Therefore, you can use all methods on `Normaliz` object. For example,

```
sage: from stdpairs import *
sage: Q=AffineMonoid(matrix(ZZ, [[0,1,1,0],[0,0,1,1],[1,1,1,1]]),
is_normaliz=True)
sage: Q.poly().hilbert_series([0,0,1])
(t + 1)/(-t^3 + 3*t^2 - 3*t + 1)
```

The method `to_macaulay2( MonomialIdeal I)` returns a dictionary storing attributes of `Macaulay2` computations. This dictionary contains an affine semigroup ring, a list of generators of an ideal, and a list of standard pairs in `Macaulay2`. For example,

```
sage: from stdpairs import *
sage: Q=AffineMonoid(matrix(ZZ,[[0,1,1,0],[0,0,1,1],[1,1,1,1]]))
sage: I=MonomialIdeal(matrix(ZZ,[[2,2,2],[0,1,2],[2,2,2]]),Q)
```

```
sage: S=to_macaulay2(I)

sage: S
{'AffineSemigroupRing': ZZ[c, a*c, a*b*c, b*c]

 monomial subalgebra of PolyRing,
 'MonomialIdeal':   2 2   2   2   2 2 2
 {a c , a b*c , a b c }

 List,
 'StandardCover': {{1, {c, b*c}}, {a*c, {c, b*c}}, {a*b*c, {c, b*c}}}

 List}
```

Moreover, Macaulay2 objects AffineSemigroupRing, MonomialSubalgebra, and list of a standard cover can be accessed via macaulay.eval(string) method with strings R, I, and SC. For instance, the example below shows how to access such Macaulay2 objects:

```
sage: macaulay2.eval('R')
ZZ[c, a*c, a*b*c, b*c]

monomial subalgebra of PolyRing

sage: macaulay2.eval('I')
  2 2   2   2   2 2 2
{a c , a b*c , a b c }

List

sage: macaulay2.eval('SC')
{{1, {c, b*c}}, {a*c, {c, b*c}}, {a*b*c, {c, b*c}}}

List
```

In Macaulay2, a type MonomialSubalgebra in the Normaliz package may correspond to an affine semigroup ring. Since Normaliz has no attributes for a monomial ideal of the type MonomialSubalgebra, the ideal is stored as a list of its generators. The standard cover of I is also sent to Macaulay2 as a nested list, similar to the output of the method standardPairs in Macaulay2.

In the other direction, from_macaulay(Macaulay2 S) translates monomialSubalgebra object S of Macaulay2 into an AffineMonoid object in StdPairs. For example,

```
sage: R = macaulay2.eval('ZZ[x,y,z]')

sage: macaulay2.needsPackage('"Normaliz"')
Normaliz

sage: macaulay2.eval('S=createMonomialSubalgebra {x^2*y, x*z, z^3}')
    2        3
ZZ[x y, x*z, z ]

monomial subalgebra of ZZ[x..z]

sage: Q=from_macaulay2('S')

sage: Q
An affine semigroup whose generating set is
[[2 1 0]
 [1 0 0]
 [0 1 3]]
```

ACKNOWLEDGEMENTS. We would like to express our deepest appreciation to Laura Matusevich for conversations and helpful comments on draft versions of this paper. Also, we are grateful to Matthias Köppe for advice on using zsolve in 4ti2. Lastly, we are indebted to an anonymous reviewer for insightful comments on the documentation and distribution of StdPairs.

SUPPLEMENT. The online supplement contains version 1.0 of `StdPairs`.

REFERENCES.

[4ti2] 4ti2 team, "4ti2—A software package for algebraic, geometric and combinatorial problems on linear spaces", available at https://4ti2.github.io.

[Bruns and Herzog 1993] W. Bruns and J. Herzog, *Cohen–Macaulay rings*, Cambridge Studies in Advanced Mathematics **39**, Cambridge University Press, 1993. MR

[Bruns and Ichim 2010] W. Bruns and B. Ichim, "Normaliz: algorithms for affine monoids and rational cones", *J. Algebra* **324**:5 (2010), 1098–1113. MR Zbl

[Hoşten and Smith 2002] S. Hoşten and G. G. Smith, "Monomial ideals", pp. 73–100 in *Computations in algebraic geometry with Macaulay* 2, edited by D. Eisenbud et al., Algorithms Comput. Math. **8**, Springer, 2002. MR Zbl

[Köppe and Labbé 2019] M. Köppe and J.-P. Labbé, "The Normaliz backend for polyhedral computations", Sage module, 2019, available at https://doc.sagemath.org/html/en/reference/discrete_geometry/sage/geometry/polyhedron/backend_normaliz.html.

[Matusevich and Yu 2020] L. F. Matusevich and B. Yu, "Standard pairs for monomial ideals in semigroup rings", 2020. arXiv

[Miller and Sturmfels 2005] E. Miller and B. Sturmfels, *Combinatorial commutative algebra*, Graduate Texts in Mathematics **227**, Springer, 2005. MR Zbl

[Sturmfels et al. 1995] B. Sturmfels, N. V. Trung, and W. Vogel, "Bounds on degrees of projective schemes", *Math. Ann.* **302**:3 (1995), 417–432. MR Zbl

RECEIVED: 22 Oct 2020 REVISED: 12 Jul 2021 ACCEPTED: 10 Aug 2021

BYEONGSU YU:

byeongsu.yu@tamu.edu

Department of Mathematics, Texas A&M University, College Station, TX, United States

Journal of Software for Algebra and Geometry vol 11, no 1, 2021

Phylogenetic trees 1
Hector Baños, Nathaniel Bushek, Ruth Davidson, Elizabeth Gross, Pamela E. Harris, Robert Krone, Colby Long, Allen Stewart and Robert Walker

Software for doing computations in graded Lie algebras 9
Clas Löfwall and Samuel Lundqvist

The relative canonical resolution: Macaulay2-package, experiments and conjectures 15
Christian Bopp and Michael Hoff

The FrobeniusThresholds package for Macaulay2 25
Daniel J. Hernández, Karl Schwede, Pedro Teixeira and Emily E. Witt

Computing theta functions with Julia 41
Daniele Agostini and Lynn Chua

Decomposable sparse polynomial systems 53
Taylor Brysiewicz, Jose Israel Rodriguez, Frank Sottile and Thomas Yahl

A package for computations with sparse resultants 61
Giovanni Staglianò

ExteriorModules: a package for computing monomial modules over an exterior algebra 71
Luca Amata and Marilena Crupi

The Schur–Veronese package in Macaulay2 83
Juliette Bruce, Daniel Erman, Steve Goldstein and Jay Yang

admcycles - a Sage package for calculations in the tautological ring of the moduli space of stable curves 89
Vincent Delecroix, Johannes Schmitt and Jason van Zelm

Coding theory package for Macaulay2 113
Taylor Ball, Eduardo Camps, Henry Chimal-Dzul, Delio Jaramillo-Velez, Hiram López, Nathan Nichols, Matthew Perkins, Ivan Soprunov, German Vera-Martínez and Gwyn Whieldon

Threaded Gröbner bases: a Macaulay2 package 123
Sonja Petrović and Shahrzad Zelenberg

Standard pairs of monomial ideals over nonnormal affine semigroups in SageMath 129
Byeongsu Yu

Computations with rational maps between multi-projective varieties 143
Giovanni Staglianò