Regular Chains in BPAS

1 `RegularChain` Class Description

The regular chain classes in BPAS provide a collection of routines for solving systems of algebraic equations by means of exact methods, a.k.a. symbolic computation. The main commands for accomplishing this are the `triangularize` and `intersect` methods of the `RegularChain` class and the `intersect` method of the `ZeroDimensionalRegularChain` class. The objects of both these classes are *regular chains.* Becuase regular chains are mathematical objects that algebraically encode geometric components of the solution space, the solutions to a system of algebraic equations can be expressed as a set of regular chains. This is precisely what `triangularize` and `intersect` accomplish: for an input polynomial p (for `intersect`) or algebraic system S (for `triangularize`), the output is a description of the solution set as a collection of `RegularChain` objects.

To state clearly what the output of `triangularize` or `intersect` is, we must define the concept of a regular chain. To this end, first observe that algebraic equations act as constraints on the geometric space defined by the possible values of their variables (typically $\mathbb{C}^n$ for n variables). For a set S of algebraic equations in n variables, and with coefficients in a subfield of $\mathbb{C}$, say the field $\mathbb{Q}$ of rational numbers, the set of points in $\mathbb{C}^n$ consistent with these algebraic constraints, *i.e.*, the locus of common zeros of the equations in S, is a certain geometric object, the algebraic variety $V(S)$, in $\mathbb{C}^n$. For example, if $S = \{x^2 + y^2 - 1\}$, then $V(S)$ is the complex unit circle in $\mathbb{C}^2$. The strategy of the `intersect` algorithm is to compute the solution space by dividing it into so-called *quasi-components*, each of which is encoded by a special kind of algebraic system, called a *regular chain*, that has a particular structure.

There are two key structural properties of regular chains that allow them to encode distinct components of algebraic varieties. First of all, regular chains have a triangular structure, in the sense that for polynomials in $\mathbb{K}[x_1, \ldots, x_n]$, with $\mathbb{K}$ a field and variable ordering $x_1 < x_2 < \cdots < x_n$, the polynomials in a regular chain T are non-constant and have *pairwise distinct main variables.* This implies that each variable can be the main variable of at most one element of T. So if $p \in T$ has x_i as its main variable, the only other variables that can appear in p are in $\{x_1, x_2, \ldots, x_{i-1}\}$. Regular chains are therefore *triangular sets.* For this reason, the `RegularChain` class in BPAS inherits from the `TriangularSet` class.

Having the structure of a triangular set allows a regular chain to encode a solution set in a manner analogous to linear systems in row echelon form. In this case, linear systems can be solved by back-substitution. Suppose that a linear system has m equations and n variables. If $m = n$ (and the system is non-singular), then the solution set is simply a unique point in $\mathbb{C}^n$. If $m < n$, however, then the solution set is a parameterized linear subspace of $\mathbb{C}^n$, and if all the equations are linearly independent it has dimension $n - m$. The situation is similar for regular chains. If $m = n$ (same number of equations as variables) for a regular chain T, then the variety $V(T)$ is a set of points in $\mathbb{C}^n$. Thus, the non-linearity of the equations allows a single system T to encode many solutions, even when the solutions are points. If $m < n$, on the other hand, then the variety $V(T)$ is a complex manifold of dimension $m - n$ embedded in $\mathbb{C}^n$. You may notice that we did not mention for regular chains an analogous condition to the non-singularity of linear systems. This is because regular chains have another structural property, over being triangular, that ensures they are "non-singular" in a sense that will now be made clear.

To see what the issue is here, consider the following example. Suppose that we have a triangular set $T = \{T_1, T_2\} = \left\{x_1^2 - 1, (x_1 + 1)x_2^2 + 1\right\}$, where $T_1, T_2 \in \mathbb{Q}[x_1, x_2]$, $x_1 < x_2$. Consider T_2, which has the largest main variable, x_2, in the set. The leading coefficient of T_2 viewed as a univariate polynomial in its main variable, called the *initial* of T_2, is $x_1 + 1$. Provided that $x_1 \neq -1$, T_2 provides a valid constraint on the ambient space of T. If $x_1 = -1$, however, then the system becomes inconsistent, because T_2 asserts that $1 = 0$. So, provided we avoid this "singular" point things are fine. The problem with T, however, is that $T_1 = (x_1 + 1)(x_1 - 1)$, so T_1 includes the "singular" case, so that at $x_1 = -1$, T gives an inconsistent set of constraints.

There are issues even if the system does not become inconsistent. Consider the positive-dimensional case where

$$T = \{T_1, T_3\} = \left\{x_1^2 - 1/4, (x_1 + 1/2)x_3^2 + x_2^2 + x_1^2 - 5/4\right\},$$

where $T_1, T_3 \in \mathbb{Q}[x_1, x_2, x_3]$, $x_1 < x_2 < x_3$. The constraint $T_1 = 0$ imposes the condition that $x_1 = \pm\frac{1}{2}$. At $x_1 = 1/2$, T_3 becomes $x_3^2 + x_2^2 - 1$, a circle in the x_2x_3-plane, and a one-dimensional manifold. But, at $x_1 = -1/2$, T_3 becomes $x_2^2 - 1$, a degenerate two-point zero-dimensional case. This is another kind of "singular" case we wish to avoid. For positive dimensional regular chains, then, avoiding such "singular" cases means that the quasi-component of the chain has *unmixed dimension*, *i.e.*, the dimension is constant across all of $W(T)$.

Thus, to avoid the possibility that a triangular set can be "singular" in these ways, we must ensure that the initials of the polynomials in the set can never be zero. Let T_k be the polynomial of T with main variable x_k, if it exists, let x_i be the largest main variable of a polynomial in a triangular set T, with T non-empty, and let $T_{<i} =_{\text{def}} T \backslash T_i$. The polynomials in T generate an ideal $\langle T \rangle$ that itself generates the variety $V(T)$. To rule out the case that h_{T_i} can be zero it must certainly be the case that $h_{T_i} \notin \langle T_{<i} \rangle$, *i.e.*, h_{T_i} must not be zero modulo $\langle T_{<i} \rangle$ (we consider $T_{<i}$ and not T here because $T_{<i}$ places the constraints

on variables less than x_i, and h_{T_i} has only variables less than x_i). But this is not the only situation in which h_{T_i} can be zero on some part of $V(T_{<i})$. If there exist any polynomials in $q \in \mathbb{K}[x_1, \dots, x_n]$ such that $h_{T_i}^k \cdot q \in \langle T_{<i} \rangle$ for some $k \in \mathbb{N}$, *i.e.*, any constraints q such that either $q = 0$ or $h_{T_i} = 0$ holding guarantees that we are in $V(T_{<i})$, then there are still parts of $V(T_{<i})$ on which $h_{T_i} = 0$. In this case, h_{T_i} is a zero-divisor modulo $\langle T_{<i} \rangle$. Thus, to avoid "singular" cases, we must therefore prevent h_{T_i} from being zero or a zero-divisor modulo $\langle T_{<i} \rangle$.

Since we can repeat this reasoning for all of the initals of the polynomials T_j in a chain modulo the ideals generated by $T_{<j}$, we require that an analogous "non-singular" condition holds simultaneously on all of the initials of the polynomials in T, *i.e.*, for h_T, so that none of the initials of polynomials in T can ever be zero or a zero-divisor. The concept we need to make this precise is the *saturated ideal* of a triangular set T, denoted $\text{sat}(T)$, which is the set of polynomials $q \in \mathbb{K}[x_1, \dots, x_n]$ such that $h_T^k \cdot q \in \langle T \rangle$, $k \in \mathbb{N}$. Given that we need to avoid zeros and zero-divisors modulo an ideal, we naturally define a polynomial to be *regular* modulo an ideal $\mathcal{I}$ if it is neither zero nor a zero-divisor modulo $\mathcal{I}$. We then finally have that a triangular set $T \subset \mathbb{K}[x_1, \dots, x_n]$ is a *regular chain* if either (1) T is empty, or (2) $T_{<T_{\max}}$ is a regular chain, where $T_{\max}$ is the polynomial in T with greatest main variable, and the initial of $T_{\max}$ is regular modulo $\text{sat}(T_{<\max})$.

Since regular chains work with the ideal $\text{sat}(T)$, we ensure that the points picked out by a regular chain are in $W(T) = V(T) \setminus V(h_T)$, as pointed out above. $W(T)$ is a quasi-component because it is defined by removing a lower dimensional boundary, and hence its zero set is not in general actually a variety (not closed in the Zariski topology); its Zariski closure $\overline{W(T)}$, however, is precisely $V(\text{sat}(T)) \subseteq V(T)$.

2 triangularize

For a set F of polynomials in $\mathbb{Q}[x_1, \dots, x_n]$, which can be encoded as as `vector F` of `SparseMultivariateRationalPolynomial` objects (abbreviated with `typedef SMQP`), which have `RationalNumber` coefficients (abbreviated with `typedef RN`)), we can compute the triangular decomposition of the variety $V(F)$ by defining an empty regular chain over the ambient space defined by the variable ordering $x_1 < x_2 < \dots < x_n$ by calling

```
vector<Symbol> R = {'x_n',...,'x_2','x_1'};
RegularChain T(R);
```

and then calling

```
vector<RegularChain<RN,SMQP>> dec;
dec = T.triangularize(F);
```

For example, to compute the intersection of the unit sphere $p_1 = x^2 + y^2 + z^2 - 1 \in \mathbb{Q}[x, y, z]$ and the unit circle $p_2 = x^2 + y^2 - 1 \in \mathbb{Q}[x, y]$, with $z < y < x$, in the ambient space $\mathbb{C}^3$ with Cartesian coordinates x, y, z, then we can use

```
vector<SMQP> F = {SMQP("x^2+y^2+z^2-1"),SMQP("x^2+y^2-1")};
vector<Symbol> R = {'x','y','z'};
RegularChain<RN,SMQP> T(R);
vector<RegularChain<RN,SMQP>> dec;
dec = T.triangularize(F);
for (auto d : dec)
   d.display();
```

which produces the output

```
 /
 | x^2 + y^2 - 1 = 0
<
 | z = 0
 \
```

which is a regular chain that picks out the complex unit circle in $\mathbb{C}^3$, described as the intersection of the unit cylinder ($x^2 + y^2 - 1 = 0$) and the xy-plane ($z = 0$).

Note that the `triangularize` method can also be called with a non-empty regular chain T. In this case it will compute the intersection of the zero set of the set F of input polynomials and the quasi-component of T. This is the sort of case handled by the method `intersect`.

3 intersect

The method `intersect` of the `RegularChain` class is essentially a special case of `triangularize` for a single polynomial input (or contrariwise, and more acurately, `triangularize` is really just a wrapper for `intersect`). For a polynomial $p \in \mathbb{Q}[x_1, \ldots, x_n]$, encoded as an `SMQP` object `p`, and a regular chain T over the ambient space defined by the variable ordering $x_1 < x_2 < \cdots < x_n$, encoded as a `RegularChain<RN,SMQP>` object `T`, we can compute the intersection of the variety $V(p)$ and the quasi-component $W(T)$ by calling

```
vector<RegularChain<RN,SMQP>> dec;
dec = T.intersect(p);
```

For example, suppose that T is the result of the example for `triangularize` above. Then we can compute the intersection of the complex unit circle and the line $x = y$ with the code

```
dec[0].upper(Symbol("z"),T);
```

which defines T to be the regular chain formed by removing the z-component from the previous result (since our present computation is really in the complex xy-plane), followed by the code

```
SMQP p("x-y");
dec = T.intersect(p);
for (auto d : dec)
   d.display();
```

which produces the output

```
 /
 | x - y = 0
<
 | 2*y^2 - 1 = 0
 \
```

so that the intersection is just the points $(x, y) = \left(\pm 1/\sqrt{2}, \pm 1/\sqrt{2}\right)$, as expected.

4 regularize

The method `regularize` is a routine that will take a polynomial p and a regular chain T and decompose T into regular chains of two types: components on which p is regular modulo $\mathrm{sat}(T)$, the regular case; and components on which p is zero modulo $\mathrm{sat}(T)$, the singular case. The return type of `regularize` is a `vector` of `PolyChainPair<PolyType,RegularChainType>` objects. If `A` is a `PolyChainPair` object, then we can access the polynomial as `A.poly` and the regular chain as `A.chain`. For the singular components returned by `regularize`, `A.poly` is zero, and for the regular components it is non-zero.

For example, suppose that a regular chain T has two polynomials, $T_3 = x^2 + y^2 - z$, describing an elliptic paraboloid, and $T_2 = y(y-1)$, describing a parabolic cylinder. Then suppose that we want to determine the regular and singular components of T for $p = xy$, a saddle surface. Then we can do so with the following code:

```
vector<Symbol> R = {'x','y','z'};
T = RegularChain<RN,SMQP>(R); // Empty chain with ordered ring R
T += SMQP("x^2+y^2-z");
T += SMQP("y*(y-1)");
p = SMQP("x*y");
vector<PolyChainPair<SMQP,RegularChain<RN,SMQP>>> components;
components = T.regularize(p);
cout << "Regular Components" << endl;
for (auto c : components) {
   if (!c.poly.isZero())
      c.chain.display();
}
cout << "Singular Components" << endl;
for (auto c : components) {
   if (c.poly.isZero())
      c.chain.display();
}
```

which produces the output

```
Regular Components
 /
 | x^2 - z + 1 = 0
<
 | y - 1 = 0
 \
Singular Components
 /
 | x^2 - z = 0
<
 | y = 0
 \
```

Thus, $p = xy$ is regular on the parabola $z = x^2 + 1$ in the plane $y = 1$ and singular on the parabola $z = x^2$ in the xz plane.