A SAGE IMPLEMENTATION OF DRINFELD'S ARGUMENTS, AND SOME VARIATIONS

by

David Bourqui & Julien Sebag

Let us recall the following theorem, due to to M. Grinberg and D. Kazhdan in case the base field is of characteristic 0 and to V. Drinfeld in general (see [**5, 4**] and also [**1**]).

Theorem 0.1. — *Let k be a field. Let V be a k-variety, with no connected component isomorphic to* $\operatorname{Spec}(k)$. *Let $\gamma \in \mathscr{L}_\infty(V)(k)$ be a rational point of the associated arc scheme, not contained in $\mathscr{L}_\infty(V_{\mathrm{sing}})$. If $(\mathscr{L}_\infty(V))_\gamma$ denotes the formal neighborhood of the k-scheme $\mathscr{L}_\infty(V)$ at the point γ, there exists an affine k-scheme S of finite type, with $s \in S(k)$, and an isomorphism of formal k-schemes:*

$$\mathscr{L}_\infty(V)_\gamma \cong S_s \hat{\otimes}_k k[[(T_i)_{i\in\mathbf{N}}]]. \tag{1}$$

1. A basic SAGE code in the case of affine plane curves

In [**1**], we show that it transpires from a detailed analysis of Drinfeld's arguments that they provide an explicit procedure for computing a pair (S, s) realizing isomorphism (1), once one has chosen an embedding of an affine neighborood of $\gamma(0)$ into an explicitely presented complete intersection and once one knows explicitely a suitable truncation of the arc γ.

In this section, we illustrate this by providing a SAGE code ([**3**]) which computes a suitable presentation of a pointed affine k-scheme (S, s) realizing isomorphism (1) in case V is an affine plane curve defined by a polynomial $F \in k[X, Y]$. In fact, we shall implement a slightly modified version of the algorithm suggested by Drinfeld's arguments which is somewhat better suited for effective computation.

More precisely, using the same notations as in section 4 of [**1**], let us write

$$\tilde{x}(T) = \tilde{x}_1(T)T^d + \tilde{x}_0(T)$$

2000 ***Mathematics Subject Classification.*** — 14E18,14B05.
Key words and phrases. — Arc scheme, curve singularity.

where $\tilde{x}_1(T), \tilde{x}_0(T) \in (k[T]_{\leq d-1})^N$. Now for any test-ring A we consider the set $\mathcal{B}'(A)$ whose elements are of the form

$$(z_A(T), \tilde{x}_{0,A}(T), \tilde{x}_{1,A}(T), \tilde{y}_A(T), q_A(T))$$

in the set $A[[T]]^N \times A[T]_{\leq d-1}^N \times A[T]_{\leq d-1}^N \times A[T]_{\leq d-1} \times \mathscr{W}(A, d)$ and satisfy the relations:

$$\begin{cases} z_A(T) = z(T) \mod \mathfrak{M}_A[[T]]; \\ \tilde{x}_{0,A}(T) = \tilde{x}_1(T) \mod \mathfrak{M}_A[T]_{\leq d-1}; \\ \tilde{x}_{1,A}(T) = \tilde{x}_0(T) \mod \mathfrak{M}_A[T]_{\leq d-1}; \\ \tilde{y}_A(T) = y(T) \mod \langle T^d, \mathfrak{M}_A \rangle; \\ q_A(T) \quad \text{divides} \quad (\partial_Y F)(\tilde{x}_{0,A}(T), \tilde{y}_A(T)); \\ q_A(T)^2 \quad \text{divides} \\ q_A(T) \left(\sum_{1 \leq i \leq N} \tilde{x}_{1,A}(T)^{(i)} (\partial_{X_i} F)(\tilde{x}_{0,A}(T), \tilde{y}_A(T)) \right) + F(\tilde{x}_{0,A}(T), \tilde{y}_A(T)), \end{cases}$$

where $\tilde{x}_{1,A}(T)^{(i)}$ is the i-th component of $\tilde{x}_{1,A}(T)$. Using Taylor's formula, it is easy to see that the map which associates with

$$(z_A(T), \tilde{x}_{1,A}(T), \tilde{x}_{0,A}(T), \tilde{y}_A(T), q_A(T))$$

the element

$$(z_A(T), q_A(T)\, \tilde{x}_{1,A}(T) + \tilde{x}_{0,A}(T), \tilde{y}_A(T), q_A(T))$$

is a natural bijection $\mathcal{B}'(A) \to \mathcal{B}(A)$. The conditions defining $\mathcal{B}'(A)$ have the computational advantage to depend only linearly on $\tilde{x}_{1,A}(T)$, that is to say, on the "higher order coefficient" of $\tilde{x}_A(T)$. Let us emphasize that even with this modification the algorithm c does not seem very efficient. In the case of the affine cusp $X^3 = Y^2$, the computation is very fast, even over the rational field. Other cases, including the plane curve given by $X^5 = Y^3$, take much more time, even over finite fields, and by increasing the multiplicity, things turn even worse. For example, on our computer and over $k = \mathbf{F}_7$, the computation took less than 0.1 second for the plane curve defined by $X^3 = Y^2$ and approximatively 40 minutes with $X^5 = Y^2$. With $X^4 = Y^3$, the computation is not finished after 12 hours.

Here is the SAGE code. The arguments are, still using the same notations as before, the polynomial F, the contact order d, and the truncated Puiseux expansions $\tilde{x}_0(T)$, $\tilde{x}_1(T)$ and $y(T) \mod T^d$, denoted in the code respectively by `F`, `cont_ord` `puiseux_X_0`, `puiseux_X_1` and `puiseux_Y`. The output is the ideal of relations `I` defining the k-scheme S.

```
# An implementation of a slightly modified version
# of Drinfeld's algorithm for plane curves

# field characterisic

p = 7
field = GF(p)
# field = QQ

R.<X,Y,T>=field[]
```

```
F = X^3-Y^2
cont_ord = 3
puiseux_X_0 = T^2
puiseux_X_1 = 0
puiseux_Y = 0


variables = [ 'x%i' % i for i in [0..cont_ord-1] ]
variables = variables + [ 'xx%i' % i for i in [0..cont_ord-1] ]
variables = variables + [ 'y%i' % i for i in [0..cont_ord-1] ]
variables = variables + [ 'q%i' % i for i in [0..cont_ord-1] ]
R1 = PolynomialRing(field,variables)
variables = variables + ['T','u','X','Y']
R2 = PolynomialRing(field,variables)
R2.inject_variables()
F=F.substitute({R.0:X})
puiseux_X_0=puiseux_X_0.substitute({R.2:T})
puiseux_X_1=puiseux_X_1.substitute({R.2:T})
puiseux_Y=puiseux_Y.substitute({R.2:T})


x=puiseux_X_0+sum([R2.gen(i)*T^i for i in [0..cont_ord-1]])
xx=puiseux_X_1+sum([R2.gen(i+cont_ord)*T^i for i in [0..cont_ord-1]])
y=puiseux_Y+sum([R2.gen(i+2*cont_ord)*T^i for i in [0..cont_ord-1]])
q=T^(cont_ord)+sum([R2.gen(i+3*cont_ord)*T^i for i in [0..cont_ord-1]])

Fxy=F.subs(X=x,Y=y)
div_deg=Fxy.degree(T)-cont_ord

# the following new variables will be used
# when dealing with the condition "q(T) divides F(x(T),y(T))"

variables = variables + [ 'p%i' % i for i in [0..div_deg] ]
R3 = PolynomialRing(field,variables)
R3.inject_variables()
x=x.substitute({R2.0:x0})
xx=xx.substitute({R2.gen(cont_ord):xx0})
y=y.substitute({R2.gen(2*cont_ord):y0})
q=q.substitute({R2.gen(3*cont_ord):q0})
F=F.substitute({R2.gen(4*cont_ord+2):X})
Fxy=F.subs(X=x,Y=y)
dXF = F.derivative(X)
dYF = F.derivative(Y)
dYFxy=dYF.subs(X=x,Y=y)
```

```
dXFxy=dXF.subs(X=x,Y=y)

# Computation of the ideal defined by the conditions
# q(T) divides (\partial_Y F)(x(T),y(T))
# and
# q(T) divise F(x(T),y(T))
# and
# q(T) divise xx(T)*(\partial_X F)(x(T),y(T))+F(x(T),y(T))/q(T)

# First step:
# q(T) divides (\partial_Y F)(x(T),y(T))

N = q.degree(T)
qq = T^N-q
rem = dYFxy
while rem.degree(T)>N-1:
L1 = [rem.coefficient(T^n)*T^(n-N)*u for n in range(N, rem.degree(T)+1)]
L2 = [rem.coefficient(T^n)*T^n for n in range(0, N)]
L2[0] = rem.substitute({T:0})
rem = sum(L1)+sum(L2)
rem = rem.substitute({u:qq})
L=[rem.coefficient(T^n) for n in range(0, rem.degree(T)+1)]
L[0] = rem.substitute({T:0})
I=R1.ideal(L)

# Second step:
# q(T) divides F(x(T),y(T))
# and computation of the quotient

pp=sum([R3.gen(i+4*cont_ord+4)*T^i for i in [0..div_deg]])
h = expand (Fxy-pp*q)
L = [h.coefficient(T^n) for n in range (0,h.degree(T)+1)]
L[0] = h.substitute({T:0})

# one eliminates the p_i

ppp = [0 for n in range (0,div_deg+1)]
for i in [div_deg..0, step=-1]:
    numer = -L[h.degree(T)+i-div_deg].substitute({R3.gen(i+4*cont_ord+4):0})
    denom = L[h.degree(T)+i-div_deg].coefficient(R3.gen(i+4*cont_ord+4))
    ppp[i] = numer/denom
    L = [L[n].substitute({R3.gen(i+4*cont_ord+4):ppp[i]})
    \ for n in range(0, h.degree(T)+1)]
I=I+R1.ideal(L)
```

```
# computation of the quotient F(x(T),y(T))/q(T)

quotient = sum([ppp[i]*T^i for i in range(0,div_deg+1)])

# Third step:
# q(T) divides xx(T)*dF/dx(x(T),y(T))+F(x(T),y-T))/q(T)

q=T^(cont_ord)+sum([R3.gen(i+3*cont_ord)*T^i for i in [0..cont_ord-1]])
N = q.degree(T)
qq = T^N-q
rem = xx*dXFxy+quotient
while rem.degree(T)>N-1:
L1 = [rem.coefficient(T^n)*T^(n-N)*u for n in range(N, rem.degree(T)+1)]

L2 = [rem.coefficient(T^n)*T^n for n in range(0, N)]
L2[0] = rem.substitute({T:0})
rem = sum(L1)+sum(L2)
rem = rem.substitute({u:qq})
L=[rem.coefficient(T^n) for n in range(0, rem.degree(T)+1)]
L[0] = rem.substitute({T:0})
I=I+R1.ideal(L)
```

2. An alternative and more efficient code for the generalized cusps

In this section we present an alternative code in the case of the curve $\mathscr{C} = \{X^N = Y^M\}$ and of the arc $\gamma(T) = (T^{\mu M}, T^{\mu N})$ where the integers $N > M \geq 2$ are coprime integers. This code is based on results of [**2**]. Its computational efficiency is much better than the code presented in the previous section. Moreover, in case $\mu = 1$, i.e. for primitive arcs it allows an explicit computation of the nilpotency index $m_\gamma(\mathscr{C})$ (see *op. cit.* for more details). Using this code, on our computer and over $k = \mathbf{F}_7$, the computation took less than 0.4 seconds for the curve singularities $X^5 = Y^2$ and $X^4 = Y^3$ (compare with the values obtained with the previous code)

The arguments are the integers `M` and `N`, and the multiplicity `mu`. The output is the ideal of relations `I1`=`I2`=I defining the affine k-scheme S. The ideals `I1` and `I2` are produced by two differents methods but coincides. The first method, producing `I1`, is less efficient but has the advantage of imposing less restriction on the characteristic.

In case `mu`=1, the nilpotency index $m_\gamma(\mathscr{C})$ turns out to be equal to the smallest integer n such that $(\sqrt{I})^n \subset I$, which is computable using SAGE or an other suitable computer algebra system.

```
# An implementation of an
# alternative to Drinfeld's algorithm
# for generalized cusps

# field characterisic
```

```
p=23
field = GF(p)
field = QQ

N = 5
M = 4
mu = 1

#############################################
## First method (p must be greater than M) ##
#############################################

h = expand(y^M-x^N)
L = [h.coefficient(T^i) for i in range(0, mu*N*M)]
L[0] = h.substitute({T:0})

# one eliminates the y_i in L

L_elim=L

for i in [mu*N-2..0, step=-1]:
    numer = L_elim[mu*N*M-mu*N+i].substitute({R.gen(i):0})
    denom = L_elim[mu*N*M-mu*N+i].coefficient(R.gen(i))
    expr = -numer/denom
    L_elim = [L_elim[n].substitute({R.gen(i):expr}) for n in range(0, mu*N*M)]

R1 = PolynomialRing(field,[ 'x%i' % i for i in [0..mu*M-2] ],order='invlex')

I1=R1.ideal(L_elim)


#####################################################
## Second method (p must be greater than mu*M*N) ##
#####################################################

g = expand(M*diff(y,T)*x-N*diff(x,T)*y)
K = [g.coefficient(T^i) for i in range(0, mu*(N+M)-1)]
K[0] = g.substitute({T:0})

# one eliminates the y_i in K

K_elim=K
for i in [mu*N-2..0, step=-1]:
     numer = K_elim[mu*M-1+i].substitute({R.gen(i):0})
```

```
    denom = K_elim[mu*M-1+i].coefficient(R.gen(i))
    expr = -numer/denom
    K_elim = [K_elim[n].substitute({R.gen(i):expr}) for n in range(0, mu*(N+M)-1)]

I2=R1.ideal(K_elim)
```

References

[1] D. Bourqui and Julien Sebag, *Drinfeld-Grinberg-Kazhdan's theorem and singularity theory*, preprint 2015.

[2] ———, *Nilpotency in arc schemes of plane singular curves*, preprint 2015.

[3] The Sage Developers, *Sage Mathematics Software (Version 6.9)*, 2015, http://www.sagemath.org.

[4] Vladimir Drinfled, *On the Grinberg-Kazhdan formal arc theorem*, preprint.

[5] M. Grinberg and D. Kazhdan, *Versal deformations of formal arcs*, Geom. Funct. Anal. **10** (2000), no. 3, 543–555.

David Bourqui, Institut de recherche mathématique de Rennes, UMR 6625 du CNRS, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes cedex (France)
E-mail : david.bourqui@univ-rennes1.fr

Julien Sebag, Institut de recherche mathématique de Rennes, UMR 6625 du CNRS, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes cedex (France)
E-mail : julien.sebag@univ-rennes1.fr