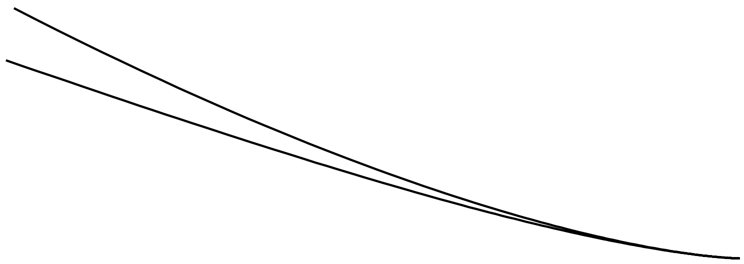
Hamburger-Noether-Entwicklung von Kurvensingularitäten

Diplomarbeit
von
Martin Lamm



Universität Kaiserslautern
Fachbereich Mathematik
Betreuer: Prof. Dr. Gert-Martin Greuel August 1999

Kommer du?
— Utenom!
Hva?
— Du får vente.
Her er mørkt, og jeg har noe tungt å hente.
Bi; jeg skal hjelpe; vi byrden skal dele.
— Nei, stå der du står! Jeg får bære det hele.
Men ikke for langt, du!
— Vær tålsom, jente;
langt eller kort, — du får vente.
Ja, vente!

Henrik Ibsen: Peer Gynt, tredje handling

Die Kurve auf der Titelseite ist das (approximierte) reelle Bild der algebraischen Menge, die durch die Gleichung

$$p_3 = y^8 + 4x^3y^6 + 6x^6y^4 + 2x^5y^5 + 4x^9y^2 + 4x^8y^3 + x^{12} + 2x^{11}y + 2x^{10}y^2 + x^{13} = 0$$

gegeben ist, in einer kleinen Umgebung des singulären Punktes $(0,0)$ (vgl. Abschnitt 6.1.1 ab Seite 177). Die Anfangsterme ihrer Parametrisierung, mit deren Hilfe das Bild gezeichnet wurde, nämlich

$$\begin{aligned} x(t) &= -t^8 + 10t^9 - 70t^{10} + 412\,t^{11} + \ldots \\ y(t) &= t^{12} - 15t^{13} + 142\,t^{14} - 1072\,t^{15} + \ldots \end{aligned}$$

wurden mit Hilfe der Hamburger-Noether-Entwicklung berechnet.

Ich danke Herrn Professor G.-M. Greuel für das interessante Thema und die Betreuung dieser Arbeit. Außerdem bedanke ich mich bei meinen Eltern für ihre geduldige Unterstützung, durch die ich mich ganz auf diese Arbeit konzentrieren konnte. Ein weiterer Dank geht an die Entwickler von Singular, des Computeralgebrasystems, auf dem der Programmierteil der Diplomarbeit aufbauen konnte.

Inhaltsverzeichnis

Einleitung

Diese Arbeit befaßt sich mit ebenen Kurvensingularitäten. Unter einer ebenen Kurve kann man sich eine eindimensionale Menge von Punkten vorstellen, die sich in einen zweidimensionalen Raum einbetten läßt. Als zweidimensionalen Raum wählen wir beispielsweise die affine Ebene $\mathbb{A}^2_k := \{(\mathrm{x},\mathrm{y}) \mid \mathrm{x},\mathrm{y} \in k\}$ über einem Körper k. Im wesentlichen hat man zwei Möglichkeiten, eine Kurve zu beschreiben:

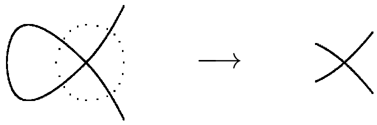
- durch eine Gleichung: Eine (affine) ebene algebraische Kurve ist die Menge $C \subset \mathbb{A}^2_k$ aller Nullstellen eines Polynoms $f \in k[x,y]$.
- durch eine oder mehrere Parametrisierungen: Eine Parametrisierung ist eine (nichtkonstante, stetige) Abbildung $\phi : k \to \mathbb{A}^2_k$, so daß $\phi(k) \subseteq C$ gilt. C heißt *irreduzibel*, falls es eine surjektive Parametrisierung besitzt. Ist C eine ebene algebraische Kurve, läßt es sich durch Polynomfunktionen parametrisieren.

Ein Punkt p der Kurve heißt *singulär*, wenn $f(p) = \frac{\partial f}{\partial x}(p) = \frac{\partial f}{\partial y}(p) = 0$ ist. Uns interessiert eigentlich nur, wie die Kurve in der Umgebung dieses Punktes aussieht, darum betrachten wir den Keim der Kurve in p (zwei Kurven haben den gleichen Keim, wenn sie in einer hinreichend kleinen Umgebung von p übereinstimmen). Einen Kurvenkeim in einem beliebigen Punkt von C bezeichnen wir als (algebraische) *Kurvensingularität* (auch wenn p nicht singulär ist – für eine solche „Singularität“ interessieren wir uns natürlich weniger).

In der Algebraischen Geometrie hat es sich als zweckmäßig erwiesen, Kurven durch Ringe darzustellen, weil man dann zu ihrer Untersuchung die Methoden der Kommutativen Algebra zur Verfügung hat. Im Fall der ebenen algebraischen Kurve ist das der Quotient aus dem Polynomring, in dem f lebt, und dem von f erzeugten Ideal: $R := k[x,y] \,/\, \langle f \rangle$. Eine algebraische Kurve ist aus algebraischer Sicht nun das Spektrum dieses Ringes $\mathcal{C} = \operatorname{Spec} R$; als Menge gesehen besteht $\mathcal{C}$ aus den Primidealen von R, und es gibt eine 1:1-Beziehung zwischen den Punkten von $C = \big\{(\mathrm{x},\mathrm{y}) \in \mathbb{A}^2_k \bigm| f(\mathrm{x},\mathrm{y}) = 0\big\}$ und den Elementen von $\mathcal{C}$, die ein maximales Ideal von R sind.

Die Kurvensingularität in einem Punkt $p = (p_x, p_y)$ erhalten wir, indem wir diesen Ring lokalisieren: $\mathcal{C}_p = \operatorname{Spec} R_p$, $R_p = \big(k[x,y]_{\langle x-p_x, y-p_y\rangle}\big) \,/\, \langle f \rangle$

Algebraische Kurvensingularitäten sind uns aber noch nicht „lokal“ genug, es kann z. B. passieren, daß sich eine irreduzible algebraische Kurve selbst schneidet. In diesem Fall sieht sie lokal so aus wie zwei sich schneidende Kurven, die Kurvensingularität ist aber dennoch irreduzibel.



Diese Möglichkeit gibt es bei *algebroiden* Kurven nicht mehr. Hier transformieren wir zuerst unser f, so daß der uns interessierende Punkt p in den Ursprung $(0,0)$ des Koordinatensystems wandert und nehmen statt der Lokalisierung die Komplettierung von R: $\widehat{R} = k[[x,y]] \,/\, \langle f \rangle$. Dabei bezeichnet $k[[x,y]]$ einen bivariaten Ring von formalen Potenzreihen. Weil $\widehat{R}$ genau ein maximales Ideal hat (das geometrisch dem Punkt $(0,0)$ entspricht), hat eine algebroide Kurve $\operatorname{Spec}\widehat{R}$ genau eine Singularität. Die unterscheidet sich nicht mehr von der Kurve selbst, denn $\widehat{R}_{(0,0)} \cong \widehat{R}$. Diese Objekte, also ebene algebroide Kurvensingularitäten, sind der Gegenstand der vorliegenden Arbeit.

Beim Studium der Kurvensingularitäten stellen sich nun zwei grundlegende Fragen:

(a) Wie wechselt man von einer Darstellung der Kurve in die andere?

Ist die Kurve durch eine Parametrisierung $(x(t), y(t))$ gegeben, kann man verhältnismäßig einfach eine geeignete Gleichung finden durch Elimination der Variable t: Man muß nur einen Erzeuger des Ideals

$$\langle x - x(t),\ y - y(t)\rangle \cap k[x,y] \quad \text{bzw.} \quad \langle x - x(t),\ y - y(t)\rangle \cap k[[x,y]]$$

finden; mit einem Computeralgebrasystem ist das zumindest für algebraische Kurven konkret möglich (bei reduziblen Kurven multipliziert man die erhaltenen Erzeuger).

Aus einer Gleichung die Parametrisierung(en) einer Kurve zu berechnen ist dagegen weit schwieriger.

(b) Sind zwei Kurvensingularitäten „äquivalent“? Um diese Frage zu beantworten, hat man eine Reihe von Invarianten eingeführt, an denen sich auch einige wichtige Eigenschaften der Kurve ablesen lassen.

Bereits Newton hat zur Lösung dieser beiden Probleme ein Verfahren entwickelt, um aus einer Gleichung für die Kurve eine Parametrisierung in einer ganz speziellen Form zu berechnen, der *Puiseuxentwicklung* (nachzulesen bei Brieskorn [BK]):

$$x(t) = t^n, \qquad y(t) = \sum_{i=m}^{\infty} a_i\, t^i$$

Viele Kurveninvarianten (z. B. die Puiseux-Paare) sind mit Hilfe dieser Entwicklung definiert und lassen sich leicht angeben, wenn man die Puiseuxentwicklung erst kennt – Jochen Fink hat das in seiner Diplomarbeit [Fi] getan.

Das Verfahren von Newton funktioniert jedoch nur für Kurven über Körpern der Charakteristik Null (wie z. B. komplexe Kurven); Kurven über anderen Körpern besitzen im allgemeinen gar keine Puiseuxentwicklung.

Diesem Problem kann man begegnen, indem man anstelle der Puiseuxentwicklung die *Hamburger-Noether-Entwicklung* (HNE) verwendet. Falls die algebroide Kurve $\mathcal{C} = \operatorname{Spec} \widehat{R}$ irreduzibel ist, ist eine HNE von $\mathcal{C}$ ein Schema

$$\begin{array}{rcl}
z_{-1} & = & a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\
z_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{r-1} & = & a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots\cdots\cdots\cdots
\end{array}$$

von Elementen $z_{-1}, z_0, \ldots, z_r$ aus dem ganzen Abschluß $\overline{\widehat{R}}$ von $\widehat{R}$. Reduzible Kurven muß man zuerst in ihre irreduziblen Komponenten (Zweige) zerlegen, bevor man eine HNE angeben kann. Anders als die Puiseuxentwicklung existiert eine HNE für jede irreduzible Kurve über einem algebraisch abgeschlossenen Körper, egal welche Charakteristik er hat.

Ausgehend von einer HNE erhält man durch Rückwärtseinsetzen eine Parametrisierung

$$x(z_r) = z_0(z_r), \qquad y(z_r) = z_{-1}(z_r)$$

und weil es Algorithmen gibt (z. B. bei Campillo [Ca]), um die HNE einer Kurve aus einer Gleichung zu berechnen, sind wir mittels Hamburger-Noether-Entwicklung in der Lage, Problem (a) zu lösen.

Analog zu den charakteristischen Exponenten der Puiseuxentwicklung besitzt auch die HNE charakteristische Größen wie z. B. die Anzahl der Zeilen und die Exponenten $h_0, h_1, \ldots, h_{r-1}$. Aus ihnen lassen sich ebenfalls die Invarianten der Kurve herleiten.

Das zu realisieren ist das Ziel dieser Arbeit: Wir wollen ausgehend von der Gleichung, durch die eine algebroide Kurve gegeben ist, für jeden ihrer Zweige eine Hamburger-Noether-Entwicklung berechnen. Mit diesen Entwicklungen wollen wir Parametrisierungen der Kurve ermitteln (dabei werden wir uns im allgemeinen auf die Anfangsterme beschränken müssen, weil die Parametrisierungen gewöhnlich Potenzreihen sind) und Invarianten bestimmen.

Das tun wir in zwei Teilen, einem theoretischen und einem praktischen. Im ersten Teil geben wir eine Einführung in Hamburger-Noether-Entwicklungen und stellen Algorithmen zu ihrer Berechnung vor. Außerdem beschäftigen wir uns mit der Frage, wie wir die Kurveninvarianten aus der HNE berechnen können und entwickeln geeignete Formeln.

Im praktischen Teil implementieren wir die Resultate aus dem ersten Teil im Computeralgebrasystem Singular (welches an der Uni Kaiserslautern entwickelt wird) in Form einer eigenen Bibliothek. Diese enthält also Prozeduren zur Berechnung von Hamburger-Noether-Entwicklungen, Invarianten und Parametrisierungen von Kurvensingularitäten. Nebenprodukt dieser Implementation sind weitere Prozeduren, die von den Algorithmen benötigte Funktionen bereitstellen, z. B. den Wechsel in algebraische Körpererweiterungen (in einer gesonderten Bibliothek), einen Quadratfreiheitstest und die Berechnung des Newtonpolygons. Die Implementierung wird anschließend anhand einiger Beispiele auf Zeit- und Speicherbedarf getestet.

Der Anhang besteht aus einem englischen Benutzerhandbuch für die beiden Bibliotheken. Anstelle eines vollständigen Listings (welches über 3500 Zeilen lang wäre) ist der Arbeit eine Diskette mit den Programmdateien beigefügt.

Teil I

Theoretischer Teil

Bevor wir die Hamburger-Noether-Entwicklung(en) einer Kurvensingularität berechnen und daraus die Invarianten, werden wir uns zuerst um eine Klärung der Begriffe bemühen: Was ist überhaupt eine algebroide Kurve, und was sind das für Invarianten, die wir berechnen wollen? Damit beschäftigen wir uns im ersten Kapitel. In diesen Zusammenhang gehört auch eine Abhandlung über das Aufblasen irreduzibler und reduzibler Kurven, auf dem die Definition der Multiplizitätensequenz (eine grundlegende Invariante) beruht.

In Kapitel 2 geht die Begriffsklärung mit der Definition von Hamburger-Noether-Entwicklungen weiter. Es gibt nicht *die* Hamburger-Noether-Entwicklung (HNE) einer Kurve, vielmehr setzt die Berechnung einer HNE zunächst die Auswahl eines Zweiges und dann die eines Erzeugendensystems des maximalen Ideals voraus. Für den Fall, daß wir eine parametrische Darstellung dieses Erzeugendensystems besitzen, geben wir einen Algorithmus zur Konstruktion der HNE an. Um die Invarianten der Singularität berechnen zu können, müssen wir für jeden Zweig eine HNE ermitteln; die Wahl des Erzeugendensystems spielt dabei keine Rolle. Wir benötigen außerdem einige Eigenschaften der HNE, um an ihr die gewünschten Informationen ablesen zu können. Diese Eigenschaften leiten wir ebenfalls her.

Kapitel 3 beschäftigt sich mit der Berechnung von Hamburger-Noether-Entwicklungen von Kurven, die durch eine Gleichung gegeben sind. Dieses Unterfangen gehen wir in zwei Etappen an: Zuerst beschränken wir uns auf irreduzible Kurven, für die bereits Antonio Campillo einen Algorithmus angegeben hat [Ca, 3.4.14]. Marc Rybowicz hat ihn in seiner Dissertation präzisiert [Ry, 1.2] und für reduzible Kurven erweitert [Ry, 1.3]. Diesen erweiterten Algorithmus stellen wir im zweiten Teil ausführlich vor. Er setzt voraus, daß der Körper, über dem gerechnet wird, algebraisch abgeschlossen ist.

In beiden Abschnitten sind die zu entwickelnden Kurven durch ein Polynom $f \in k[x,y]$ gegeben, und wie schon bemerkt liegt der HNE eine Vorauswahl zugrunde: Der Wahl eines Zweiges entspricht die Wahl eines irreduziblen Faktors $f_i \in k[[x,y]]$ von f, und ausgehend von dieser Potenzreihe wählen wir als Erzeuger des maximalen Ideals von $k[[x,y]] \,/\, \langle f_i \rangle$ die Restklassen von x und y. Eine Eigenschaft des Algorithmus von Rybowicz kommt uns sehr gelegen: Wir brauchen die Faktoren f_i gar nicht zu kennen und bekommen trotzdem für jeden von ihnen eine Hamburger-Noether-Entwicklung, die nach Vorgabe des Erzeugendensystems bis auf eine eventuell mögliche Vertauschung der Rollen von x und y eindeutig ist.

Der dritte Abschnitt berücksichtigt bereits die spätere Implementierung in Singular: Gegeben ist dann kein Polynom über einem algebraisch abgeschlossenen Körper, sondern über einem endlichen Körper, $\mathbb{Q}$ oder einer algebraischen Erweiterung von $\mathbb{Q}$. Es ist nicht möglich, in den algebraischen Abschluß dieses Körpers zu wechseln, aber wir werden sehen, daß es stets eine endliche algebraische Körpererweiterung gibt, über der f in die gleichen Faktoren zerfällt und die Kurve die gleichen Hamburger-Noether-Entwicklungen besitzt (Satz 3.26). Bei irreduziblen Kurven erfüllt der Körper, aus dem die Koeffizienten von f sind, bereits selbst diese Eigenschaft, d. h. wir brauchen am ersten Algorithmus nichts zu ändern. Beim zweiten ist dagegen im allgemeinen ein Wechsel des Körpers in eine algebraische Erweiterung nötig; wir modifizieren den Algorithmus so, daß er den Wechsel bei Bedarf selbst durchführt. Bedingt durch die unterschiedliche Darstellung von Körpererweiterungen in Singular und Scratchpad (Rybowiczs System) gehen wir bei der Implementation des Körperwechsels eigene Wege: Singular kennt nur primitive algebraische Erweiterungen, daher werden wir in 3.3 zwei Algorithmen[1] entwickeln, die das Minimalpolynom eines primitiven Elements mit Hilfe von Standardbasen ausrechnen.

In Kapitel 4 schließlich machen wir uns darüber Gedanken, wie wir aus der Hamburger-Noether-Entwicklung die Invarianten einer Kurve bzw. eines bestimmten Zweiges berechnen können. Einen Teil von ihnen werden wir direkt aus der HNE gewinnen, z. B. die Multiplizitätensequenz, die charakteristischen Exponenten, das Geschlecht, den Grad des Konduktors, die δ-Invariante und die Schnittmultiplizität zweier Zweige. Andere Invarianten lassen sich besser aus den eben aufgezählten herleiten als direkt aus der HNE, dazu zählen etwa die Puiseux-Paare (aus den charakteristischen Exponenten) und die Wertehalbgruppe (aus den Puiseux-Paaren).

[1] Der erste funktioniert für endlich viele algebraische Elemente über unendlichen Körpern, der zweite für zwei Elemente über endlichen Körpern. Die Praxis hat gezeigt, daß der erste für alle Körper zu gebrauchen ist, aber es kann bei sehr kleinen Körpern passieren, daß er kein primitives Element findet, weil er es nur unter den Linearkombinationen der gegebenen Elemente sucht.

1 Algebroide Kurvensingularitäten

Die hier kurz vorgestellten Grundlagen sind den Büchern von Kunz [Ku], Eisenbud [Ei] und Campillo [Ca] entnommen (Beweise siehe dort). Wir definieren im folgenden algebroide Kurven, Zweige, Parametrisierungen von Zweigen, quadratische Transformation und wichtige Invarianten von Kurven. Anders als Campillo betrachten wir am Rande auch Kurven über nicht algebraisch abgeschlossenen Körpern, weil wir im zweiten Teil genötigt sein werden, über solchen Körpern zu rechnen.

1.1 Grundlegende Definitionen

Definition 1.1 Sei R ein lokaler Ring mit maximalem Ideal $\mathfrak{m}$. Wir geben R eine Topologie, die sogenannte *Krull-Topologie* ($\mathfrak{m}$-adische Topologie); diese wird induziert von den Mengen $r + \mathfrak{m}^i$, $r \in R$, $i \in \mathbb{N}$. Nach dem *Krull'schen Durchschnittssatz* [Kru] ($\bigcap_{\nu \in \mathbb{N}} \mathfrak{m}^\nu = \{0\}$) ist R hausdorffsch. Eine *Cauchyfolge* in R ist damit eine Folge

$$(r_i)_{i\in\mathbb{N}} : \forall\, \nu \in \mathbb{N} \quad \exists\, N \in \mathbb{N} : \quad r_m - r_n \in \mathfrak{m}^\nu \quad \forall\, m, n \geq N$$

Analog ist die Konvergenz von $(r_i)_{i\in\mathbb{N}}$ definiert.
R heißt **komplett** (**vollständig**), wenn jede Cauchyfolge in R konvergiert.

Definition 1.2 Sei k ein beliebiger Körper. Eine **algebroide Kurve $\mathcal{C}$ über k** ist das affine Schema $\mathcal{C} = \operatorname{Spec} R$ eines reduzierten lokalen noetherschen Ringes R mit folgenden Eigenschaften:

- R ist komplett
- R hat die (Krull-)Dimension 1
- k ist Koeffizientenkörper von R, d. h. $k \subset R$ und die Einschränkung des kanonischen Epimorphismus $R \to R/\mathfrak{m}$ auf k ($\mathfrak{m}$ das maximale Ideal von R) ist ein Isomorphismus $k \cong R/\mathfrak{m}$.

R heißt dabei der **lokale Ring der Kurve $\mathcal{C}$**. Es ist auch üblich, R selbst als die algebroide Kurve zu bezeichnen, denn man rechnet in der Praxis nur mit dem lokalen Ring und nicht mit dem Spektrum. Ferner sprechen wir bei $\mathcal{C}$ oder R auch von einer (algebroiden) **Kurvensingularität**[1].

Definition 1.3 $\mathbf{Emb}(\boldsymbol{R}) := \dim_k(\mathfrak{m}/\mathfrak{m}^2)$ heißt die **Einbettungsdimension** von R (und ist stets endlich, da R noethersch). Eine Kurve R mit $\mathrm{Emb}(R) \leq 2$ heißt **eben**.

Notation: Ist R ebene Kurve, so läßt sich das maximale Ideal $\mathfrak{m}$ von R durch zwei Elemente erzeugen. Wir werden in diesem Fall schreiben:

$$R = k[\![x, y]\!]$$

wobei $x, y \in R$ die Urbilder von $X + I$, $Y + I$ unter $R \cong k[\![X, Y]\!]/I$ sind.

Satz 1.1 *Sei R der lokale Ring einer algebroiden Kurve. Dann gilt:*

(i) Struktursatz von Cohen *([Coh], Thm. 9):*
Es gibt ein Ideal $I \subset k[\![x_1, \ldots, x_n]\!]$, so daß

$$R \cong k[\![x_1, \ldots, x_n]\!]/I \tag{1.1}$$

Dabei kann $n = \mathrm{Emb}(R)$ gewählt werden; dies ist die kleinste mögliche Wahl.
Weil R reduziert ist, ist I ein Radikalideal.

(ii) Es gibt Primideale $p_1, \ldots, p_s \subset k[\![x_1, \ldots, x_n]\!]$ mit der Eigenschaft $p_i \not\subset p_j$ für $i \neq j$ und $I = \bigcap_{i=1}^{s} p_i$.
Diese entsprechen genau den minimalen Primidealen $\mathfrak{p}_1, \ldots, \mathfrak{p}_s \subset R$.
(Beweis folgt aus dem Satz über die Primärzerlegung, z. B. [Wae, §118]).

(iii) Ist R eben, so ist $I \subset k[\![x_1, x_2]\!]$ ein Hauptideal.
(Dies folgt aus [Ei, 10.5 & 3.11b] sowie Teil (ii)).

Bemerkung: Beim Studium ebener algebroider Kurven reicht es also, Ringe der Form $k[\![X, Y]\!] \,/\, \langle f \rangle$ zu betrachten, f eine quadratfreie Potenzreihe.

Wir stellen darum einige wesentliche Eigenschaften von formalen Potenzreihenringen zusammen (siehe z. B. Anhang K bei [Ku]):

Satz 1.2 *Sei k ein beliebiger Körper, $A = k[\![x_1, \ldots, x_n]\!]$ der Ring der formalen Potenzreihen über k in n Variablen. Dann gilt:*

(i) A ist noethersch (Hilbertscher Basissatz für Potenzreihenringe).

(ii) A ist komplett und hausdorffsch bzgl. $\mathfrak{m} = \langle x_1, \ldots, x_n \rangle$.

(iii) A ist ein regulärer lokaler Ring mit maximalem Ideal $\mathfrak{m}$.

(iv) A ist ein Integritätsbereich.

(v) A ist faktoriell.

[1] Wie schon in der Einleitung bemerkt, gibt es im algebroiden Fall keinen Unterschied zwischen einer Kurve und der zugehörigen Kurvensingularität.

Satz 1.1 motiviert folgende

Definition 1.4 Sei R der lokale Ring einer algebroiden Kurve $\mathcal{C}$, $\mathfrak{p} \subset R$ ein minimales Primideal, dann heißt $R/\mathfrak{p}$ ein **Zweig** der Kurve R (eigentlich: $\operatorname{Spec} R/\mathfrak{p}$ ist ein Zweig der Kurve $\mathcal{C}$).
Eine Kurve heißt **irreduzibel**, wenn sie nur einen Zweig besitzt, sonst **reduzibel**.

Bemerkung: Eine algebroide Kurve ist genau dann irreduzibel, wenn ihr lokaler Ring ein Integritätsbereich ist.

1.2 Parametrisierungen

Für den Rest dieses Kapitels nehmen wir an, daß k ein algebraisch abgeschlossener Körper ist.

Definition 1.5 Sei R ein beliebiger Ring. Die Lokalisierung von R nach der Menge der Nichtnullteiler von R heißt der **volle Quotientenring** von R und wird mit $\boldsymbol{Q(R)}$ bezeichnet. Mit $\overline{\boldsymbol{R}}$ ist der ganze Abschluß von R in $Q(R)$ gemeint.

Ist R der lokale Ring einer irreduziblen algebroiden Kurve über einem *algebraisch abgeschlossenen* Körper k, so ist $\overline{R}$ ein (lokaler) diskreter Bewertungsring. Genauer: Wenn $\overline{\mathfrak{m}}$ das maximale Ideal von $\overline{R}$ ist, $t \in \overline{\mathfrak{m}} \backslash \overline{\mathfrak{m}}^2$ beliebig und $k[\![T]\!]$ ein univariater formaler Potenzreihenring, dann ist durch $T \mapsto t$ ein Isomorphismus $k[\![T]\!] \cong \overline{R}$ von k-Algebren definiert ([Ca, 1.3.1]). Es gilt folglich

$$\overline{R} = k[\![t]\!], \quad Q(R) = k(\!(t)\!)$$

Ein $t \in \overline{R}$ mit $\overline{R} = k[\![t]\!]$ heißt **regulärer Parameter** der Kurve R.

Es bezeichne $\boldsymbol{\underline{\nu}}$ die natürliche Bewertung von $k(\!(t)\!)$. Wir nennen $\underline{\nu} = \underline{\nu}_R$ die zu R **assoziierte Bewertung**. Für $r \in \overline{R}$ beliebig gibt es ein $f \in k[\![T]\!]$, so daß $f(t) = r$. Es gilt dann $\underline{\nu}(r) = \operatorname{ord}_T(f)$, wobei $\operatorname{ord}(f) = \operatorname{ord}_T(f)$ die Ordnung der Potenzreihe f bezeichnet.

Definition 1.6 $0 \neq z \in R$ heißt **Parameter** der Kurve R, wenn das Hauptideal $\langle z \rangle$ $\mathfrak{m}$-primär ist. Ist R irreduzibel, so heißt ein Parameter z **transversal**, wenn er minimalen Wert $\underline{\nu}(z)$ unter den Elementen von $\mathfrak{m}$ hat.

Bemerkung: Für irreduzible Kurven R ist jedes Element von $\mathfrak{m} \setminus \{0\}$ ein Parameter, denn R ist nullteilerfrei und $\operatorname{Spec} R = \{(0), \mathfrak{m}\}$.

Definition 1.7 Sei R der lokale Ring einer irreduziblen algebroiden Kurve, $\mathfrak{m} \subset R$ das maximale Ideal, $\{y_1, \ldots, y_N\} \subset \mathfrak{m}$ ein minimales Erzeugendensystem von $\mathfrak{m}$ (beachte $N = \operatorname{Emb}(R)$). Eine **lokale parametrische Darstellung** der Kurve R **in der Basis** $(y_1, \ldots, y_N)$ ist ein Tupel von Potenzreihen

$$\Big(x_1(T), \ldots, x_N(T)\Big) \in \bigoplus_{i=1}^{N} k[\![T]\!]$$

für das es eine endliche Körpererweiterung $L \supseteq Q(R)$ und ein $u \in L$ gibt, so daß

$$\forall\, i = 1, \ldots, N : \qquad y_i = x_i(u) \tag{1.2}$$

Ein Tupel $\big(x_1(T), \ldots, x_N(T)\big)$ heißt (lokale) **Parametrisierung** von R, wenn es ein minimales Erzeugendensystem $\{y_1, \ldots, y_N\}$ von $\mathfrak{m}$ gibt, so daß $\big(x_1(T), \ldots, x_N(T)\big)$ lokale parametrische Darstellung von R in der Basis $(y_1, \ldots, y_N)$ ist.

Eine **Parametrisierung** einer *beliebigen* algebroiden Kurve R ist die Parametrisierung eines Zweiges von R.

Eine Parametrisierung (bzw. lokale parametrische Darstellung) heißt **primitiv**, wenn u in Gleichung (1.2) in $Q(R)$ selbst liegt, d. h. $L = Q(R)$ gewählt werden kann.

Beispiel: Sei $\varphi : \overline{R} \cong k[[T]]$ ein Isomorphismus (s. oben), $\{y_1, \ldots, y_N\} \subset \mathfrak{m}$ ein minimales Erzeugendensystem, dann ist $(\varphi(y_1), \ldots, \varphi(y_N))$ eine primitive lokale parametrische Darstellung von R in der Basis $(y_1, \ldots, y_N)$.
Jede algebroide Kurve über einem algebraisch abgeschlossenen Körper besitzt also eine primitive Parametrisierung. Diese Aussage ist im allgemeinen für Kurven über beliebigen Körpern nicht richtig.

Bemerkung: Sei R eine ebene algebroide Kurve, d. h. $\exists\, f :\ R = k[[X,Y]]\,/\,\langle f \rangle$. Dann läßt sich zeigen, daß $\big(x(T), y(T)\big)$ genau dann eine lokale parametrische Darstellung eines Zweiges von R in der Basis $([X]_{f_i}, [Y]_{f_i})$ ist (f_i ein irreduzibler Faktor von f), wenn

$$k[[T]] \ni f(x(T), y(T)) = 0 \qquad \text{ist}^2.$$

Eine solche Parametrisierung nennen wir auch **Parametrisierung von f**.

1.3 Invarianten

Wir betrachten eine algebroide Kurve $\mathcal{C}$ über dem Körper k mit lokalem Ring R und maximalem Ideal $\mathfrak{m}$. $\overline{R}$ sei der ganze Abschluß von R in $Q(R)$. Folgende (topologische) Invarianten sind allgemein bekannt:

Definition 1.3 Die **Einbettungsdimension** von R wurde bereits vorgestellt: $\mathbf{Emb}(\boldsymbol{R}) := \dim_k(\mathfrak{m}/\mathfrak{m}^2)$

Definition 1.8 Die **Multiplizität $\boldsymbol{e(R)}$** der Kurve ist definiert als die Multiplizität des maximalen Ideals $\mathfrak{m}$, d. h. $e(R)$ ist diejenige ganze Zahl e, für die $\dim_k(R/\mathfrak{m}^n) = en + q \quad \forall\, n \gg 0$ gilt.

Definition 1.9 $\boldsymbol{\delta} = \ \delta_R\ := \ \dim_k(\overline{R}/R)$ heißt der **Singularitätengrad** von $\mathcal{C}$ (**$\boldsymbol{\delta}$-Invariante**). Analog ist der Singularitätengrad eines Zweiges $Z = R/\mathfrak{p}$ von R definiert als $\delta_Z := \dim_k(\overline{Z}/Z)$.

[2] $f(x(T), y(T))$ ist definiert als Grenzwert einer bzgl. der Krulltopologie konvergenten Reihe

Definition 1.10 Die Anzahl r der minimalen Primideale von R heißt die **Anzahl der Zweige** von $\mathcal{C}$ (oder von R).

Definition 1.11 $\boldsymbol{\mu} := 2\delta - r + 1$ heißt die **Milnorzahl** von R.

Bemerkung: Ist speziell $R = \mathbb{C}[\![X,Y]\!] \,/\, \langle f \rangle$ der lokale Ring einer ebenen algebroiden Kurve über $\mathbb{C}$, so gilt die bekanntere Formel:

$$\mu = \dim_{\mathbb{C}} \mathbb{C}[\![X,Y]\!] \,/\, \langle f_X, f_Y \rangle$$

wobei f_X bzw. f_Y die (formale) partielle Ableitung von f nach X bzw. nach Y meint (vgl. [Mi, 10.5]).

Definition 1.12 Sei $R = k[\![x,y]\!] \,/\, \langle f \rangle$ der lokale Ring einer irreduziblen algebroiden Kurve $\mathcal{C}$ und $k[\![x,y]\!]/\, \langle g \rangle$ der lokale Ring einer weiteren (nicht notwendigerweise irreduziblen) algebroiden Kurve $\mathcal{D}$. Dann heißt

$$\operatorname{mult}(\mathcal{C},\mathcal{D}) := \underline{\nu}_R(g(\mathrm{x},\mathrm{y}))$$

die **Schnittmultiplizität** von $\mathcal{C}$ und $\mathcal{D}$, wobei $\underline{\nu}_R$ die zu R assoziierte Bewertung und $\mathrm{x},\mathrm{y} \in R$ die Restklassen von x, y sind.

Bemerkung: Weil $\mathcal{C}$ irreduzibel ist, gibt es eine primitive lokale parametrische Darstellung $(x(T), y(T))$ von R in der Basis (x,y), d. h.

$\exists\, t \in \overline{R} : \mathrm{x} = x(t),\ \mathrm{y} = y(t) \ \Rightarrow\ \underline{\nu}_R(\mathrm{x}) = \operatorname{ord}(x(T)),\ \underline{\nu}_R(\mathrm{y}) = \operatorname{ord}(y(T))$

und damit hat die Schnittmultiplizität auch folgende bekanntere Form:

$$\operatorname{mult}(\mathcal{C},\mathcal{D}) = \operatorname{ord}(\, g(x(T), y(T))\,)$$

Außerdem gilt ([Ca, 2.3.2]): Ist auch $\mathcal{D}$ irreduzibel, so ist

$$\operatorname{mult}(\mathcal{C},\mathcal{D}) = \operatorname{mult}(\mathcal{D},\mathcal{C})$$

Im folgenden beschränken wir uns auf irreduzible ebene algebroide Kurvensingularitäten (das sind insbesondere die *Zweige* von *beliebigen* ebenen algebroiden Kurvensingularitäten). Es gibt also ein $f \in k[\![X,Y]\!] : R = k[\![X,Y]\!] \,/\, \langle f \rangle$.

Bemerkung: Die Multiplizität einer irreduziblen Kurve läßt sich einfacher schreiben: Sei z ein transversaler Parameter von R, dann gilt $e(R) = \underline{\nu}(z)$ [Ca, 1.4.7].

Definition 1.13 Sei R der lokale Ring einer irreduziblen ebenen algebroiden Kurve, sei $\underline{\nu}$ die zu R assoziierte Bewertung. Dann heißt

$$\boldsymbol{\Gamma} = \Gamma(R) := \underline{\nu}\,(R \setminus \{0\})$$

die **Wertehalbgruppe** (oder auch nur **Halbgruppe**) von R. Die **Erzeuger der Wertehalbgruppe** Γ bezeichnen wir mit $\boldsymbol{\overline{\beta}_0}, \ldots, \boldsymbol{\overline{\beta}_g}$. Man erhält sie wie folgt:

Wähle $\overline{\beta}_0 \in \Gamma \setminus \{0\}$ minimal.
Solange $\langle \overline{\beta}_0, \ldots, \overline{\beta}_{i-1} \rangle \neq \Gamma$ wähle $\overline{\beta}_i \in \Gamma \setminus \langle \overline{\beta}_0, \ldots, \overline{\beta}_{i-1} \rangle$ minimal.

Definition 1.14 Sei $\overline{R}$ der ganze Abschluß von R, dann heißt

$$\mathfrak{K} := \left\{ r \in \overline{R} \;\middle|\; r\overline{R} \subseteq R \right\}$$

der **Konduktor von $\overline{R}$ in R**.

Definition 1.15 und Lemma Es gibt eine positive ganze Zahl $\boldsymbol{c}$ mit der Eigenschaft: Ist $t \in \overline{R}$ regulärer Parameter der Kurve, dann ist $\mathfrak{K} = t^c \overline{R}$ (denn $\mathfrak{K}$ ist ein Ideal und $\overline{R}$ ein diskreter Bewertungsring). Dieses c heißt der **Grad des Konduktors** von $\overline{R}$ in R.

Bemerkung: Der Grad des Konduktors läßt sich direkt an der Halbgruppe ablesen ([Ca, 4.4.2]): Er ist das kleinste c mit der Eigenschaft

$$c + \mathbb{N} \subset \Gamma$$

In Charakteristik 0 kann man einer irreduziblen Kurvensingularität Invarianten zuordnen, die von der Puiseuxentwicklung abgeleitet sind. Um dies für beliebige (wohlgemerkt algebraisch abgeschlossene) Körper zu verallgemeinern, bedienen wir uns komplexer Modelle. Wir folgen der Darstellung von Campillo, der einen etwas schwächeren Äquivalenzbegriff für Singularitäten verwendet als topologische Äquivalenz, nämlich „äquiresolubel“ [Ca, chap. 3.1 & 3.2]. Zunächst müssen wir wissen, was es heißt, Singularitäten aufzublasen:

Definition 1.16 Für $x \in \mathfrak{m} \setminus \{0\}$ sei

$$x^{-1}\mathfrak{m} := \left\{ \frac{z}{x} \;\middle|\; z \in \mathfrak{m} \right\} \subset Q(R)$$

eine Teilmenge des Quotientenkörpers von R, und

$$R_x := R\left[x^{-1}\mathfrak{m}\right]$$

die von dieser Menge erzeugte R-Unteralgebra von $Q(R)$. Wähle einen transversalen Parameter $x \in \mathfrak{m}$. Dann ist die (strikte) **quadratische Transformierte** von R definiert als $\boldsymbol{R_1} := R_x$.

Lemma 1.3 *R_1 ist wieder eine irreduzible ebene algebroide Kurve über dem gleichen*[3] *Körper k und hängt nicht von der Wahl von x ab. R_1 ist ganz über R, d. h. $R \subseteq R_1 \subseteq \overline{R}$, und für die Multiplizität gilt $e(R_1) \leq e(R)$. Genau dann ist $R_1 = R$, wenn R regulär ist.*
Ist $\{x, y\}$ ein Erzeugendensystem von $\mathfrak{m}$, x transversal, dann gibt es ein $a \in k$, so daß $\{x, \frac{y}{x} - a\}$ das maximale Ideal $\mathfrak{m}_1$ von R_1 erzeugt.

Beweis: [Ca, 1.5.4, 1.5.5, 1.5.6 und 1.5.8]

Weil R_1 ebenfalls eine irreduzible algebroide Kurve ist, können wir für R_1 wieder die quadratische Transformierte berechnen. Diese nennen wir R_2, und so fort.

[3] hier geht wesentlich ein, daß k algebraisch abgeschlossen ist

Satz 1.4 (Auflösung von Singularitäten) *Ist R eine irreduzible algebroide Kurve, dann ist die aufsteigende Kette*

$$R \subseteq R_1 \subseteq \cdots \subseteq R_i \subseteq R_{i+1} \subseteq \cdots \subseteq \overline{R} \tag{1.3}$$

sukzessiver quadratischer Transformationen stationär und endet in $\overline{R}$ [Ca, 1.5.10].

Eine solche aufsteigende Kette (1.3) von Ringen heißt **Auflösungskette** von R.

Definition 1.17 Zwei (wohlgemerkt irreduzible) algebroide Kurven über dem gleichen Körper heißen **äquiresolubel** (gleichauflösbar), wenn die Multiplizitätensequenzen der Ringe in den korrespondierenden Auflösungsketten von Satz 1.4 identisch sind. Das heißt: Sie sind genau dann äquiresolubel, wenn gleich viele quadratische Transformationen notwendig sind, bis die Transformierte eine glatte Kurve ist, und wenn die Multiplizitäten, die im Verlauf der Transformationen auftreten, bei beiden identisch sind.

Definition 1.18 Unter einer **Invarianten** einer Kurve verstehen wir eine Zahl, die nur von der k-Isomorphismenklasse des zugehörigen lokalen Ringes abhängt.

Bemerkung: Die in diesem Abschnitt bisher genannten Invarianten sind allein mit Hilfe des lokalen Ringes der Kurve definiert worden und daher auch Invarianten im letzteren Sinn.

Definition 1.19 Ein **vollständiges System von Invarianten für die Gleichauflösung** ist eine Familie von Invarianten der Kurve, die eindeutig durch die Multiplizitätensequenz der Auflösungskette bestimmt ist und sie umgekehrt auch eindeutig bestimmt.

Beispiele für vollständige Systeme von Invarianten:

- Die Multiplizitätensequenz der Auflösungskette (trivialerweise)
- Die Erzeuger $\overline{\beta}_0, \ldots, \overline{\beta}_g$ von Γ (inklusive ihrer Anzahl) [Ca, 4.3.7] bzw. Korollar zu Lemma 4.15 auf Seite 135.

Definition 1.20 Sei R eine (irreduzible) ebene algebroide Kurve über dem Körper k. Ein **komplexes Modell** für R ist eine ebene algebroide Kurve $R_{\mathbb{C}}$ über dem Körper $\mathbb{C}$ der komplexen Zahlen, deren Auflösungskette die gleiche Multiplizitätensequenz besitzt wie die von R.
Das heißt, wenn

$$\begin{array}{ccccc} R \subset R_1 \subset & \cdots & \subset R_M = \overline{R} \\ R_{\mathbb{C}} \subset (R_{\mathbb{C}})_1 \subset & \cdots & \subset (R_{\mathbb{C}})_N = \overline{R_{\mathbb{C}}} \end{array}$$

die Auflösungsketten von R bzw. $R_{\mathbb{C}}$ sind, dann gilt:

$$M = N, \quad e(R) = e(R_{\mathbb{C}}) \quad \text{und} \quad e(R_i) = e((R_{\mathbb{C}})_i) \quad \forall\, 1 \leq i \leq M$$

Wir werden später mit Hilfe der Hamburger-Noether-Entwicklung zeigen, daß jede ebene algebroide Kurve über einem algebraisch abgeschlossenen Körper (beliebiger Charakteristik) ein komplexes Modell besitzt (Lemma 4.2). Damit ist der Weg frei, Invarianten komplexer Kurvensingularitäten, die eindeutig durch die Multiplizitätensequenz bestimmt sind, für Kurven über beliebigen Körpern zu verallgemeinern.

Wir kommen jetzt zu den Invarianten, die von der Puiseuxentwicklung abgeleitet sind.

Definition 1.21 Sei R eine irreduzible ebene algebroide Kurve über k, $\{x, y\}$ ein Erzeugendensystem des maximalen Ideals und $(\mathrm{x}(T), \mathrm{y}(T))$ eine lokale parametrische Darstellung von R in der Basis (x, y). $(\mathrm{x}(T), \mathrm{y}(T))$ heißt **in Puiseux-Form**, wenn es ein $n \in \mathbb{N}$ gibt mit $\mathrm{x}(T) = T^n$ oder $\mathrm{y}(T) = T^n$.
Eine **Puiseuxentwicklung** (**Puiseux-Parametrisierung**) von R ist eine lokale parametrische Darstellung von R (in irgendeiner Basis) in Puiseux-Form.

Die Existenz einer Puiseuxentwicklung wird uns in Charakteristik 0 von folgendem Satz garantiert:

Satz 1.5 (Puiseuxentwicklung) *Sei K ein algebraisch abgeschlossener Körper der Charakteristik 0, R eine irreduzible ebene algebroide Kurve über K und $\{x, y\}$ ein Erzeugendensystem des maximalen Ideales, so daß x transversal ist. Dann gibt es eine primitive parametrische Darstellung von R in der Basis (x, y) in Puiseux-Form:*

$$\mathrm{x}(T) = T^n, \qquad \mathrm{y}(T) = \sum_{i=n}^{\infty} a_i T^i$$

Beweis: [BK, III.8.3] und [Ca, 2.1.1].

Bemerkung: In Charakteristik > 0 gilt dieser Satz nicht, auch nicht in der abgeschwächten Form, daß R stets eine Puiseuxentwicklung hat.[4]

Beispiel: Betrachte $R = k[[x, y]] \subset k[[t]]$, k ein Körper der Charakteristik p, wobei

$$x := t^{p^2} + t^{p^2+1}, \qquad y := t^{p^2+p} + t^{p^2+p+1}$$

Diese Kurve besitzt keine Puiseuxentwicklung, denn es gibt kein Element in $\langle x, y\rangle \setminus \langle x, y\rangle^2$, welches sich in der Form t'^n schreiben läßt für einen regulären Parameter $t' \in k[[t]]$ und ein $n > 0$ (vgl. [Ca, 2.1.2]).

Definition 1.22 Mit den Bezeichnungen aus Satz 1.5 definieren wir (zunächst für Charakteristik 0) die **charakteristischen Exponenten $\boldsymbol{\beta_0, \ldots, \beta_g}$** einer irreduziblen algebroiden Kurve:

$$\beta_0 := n = e(R)$$

Ist $\beta_0 > 1$ (was genau dann der Fall ist, wenn R nicht regulär ist), so sei

$$\beta_1 := \min\{i \mid a_i \neq 0, \, \mathrm{ggT}(i, \beta_0) < \beta_0\}$$

[4] Dies wäre schon erfüllt, wenn es für jede Kurve ein (ausgezeichnetes) Erzeugendensystem des maximalen Ideals gäbe, welches eine Puiseuxdarstellung besitzt.

Sind $\beta_0, \ldots, \beta_\nu$ definiert und $\mathrm{ggT}(\beta_0, \ldots, \beta_\nu) > 1$, dann sei

$$\beta_{\nu+1} := \min\{i \mid a_i \neq 0,\ \mathrm{ggT}(\beta_0, \ldots, \beta_\nu, i) < \mathrm{ggT}(\beta_0, \ldots, \beta_\nu)\}$$

Weil die parametrische Darstellung *primitiv* ist, gibt es ein $g \in \mathbb{N}$, so daß $\mathrm{ggT}(\beta_0, \ldots, \beta_g) = 1$ ist (folgt aus [Ca, 2.1.13]). Damit ist die Anzahl der charakteristischen Exponenten endlich.

Satz 1.6 *(i) Die charakteristischen Exponenten hängen weder von der Wahl des Erzeugendensystems $\{x, y\}$ ab noch von der parametrischen Darstellung von R, sind also wohldefiniert. [Za1, p. 995]*

(ii) Die charakteristischen Exponenten sind ein vollständiges System von Invarianten für die Gleichauflösung *(Beweis: Satz 4.14 auf Seite 133).*

Wir definieren jetzt die charakteristischen Exponenten für Kurven in beliebiger Charakteristik:

Definition 1.23 Sei R eine irreduzible ebene algebroide Kurve über einem beliebigen Körper k und sei $R_{\mathbb{C}}$ ein komplexes Modell für R. Dann sind die **charakteristischen Exponenten** von R definiert als die charakteristischen Exponenten von $R_{\mathbb{C}}$.

Bemerkung: Die charakteristischen Exponenten sind wohldefiniert, weil per definitionem alle komplexen Modelle von R äquiresolubel sind, und wegen des letzten Satzes. Campillo hat gezeigt ([Ca, 3.2.13]), daß in dem Fall, daß R selbst eine Puiseuxentwicklung in der Form von Satz 1.5 besitzt, die so definierten charakteristischen Exponenten mit den charakteristischen Exponenten der Puiseuxreihe übereinstimmen.

Mit Hilfe der charakteristischen Exponenten sind wir in der Lage, ein weiteres vollständiges System von Invarianten für die Gleichauflösung zu definieren (vgl. Korollar zu Satz 4.14):

Definition 1.24 Sei R eine irreduzible ebene algebroide Kurve mit charakteristischen Exponenten $\beta_0, \ldots, \beta_g$. Dann definiere die **Puiseux-Paare (m_1, n_1), $\ldots, (m_g, n_g)$** von R wie folgt: Sind $(m_1, n_1), \ldots, (m_{i-1}, n_{i-1})$ bereits definiert, dann ist (m_i, n_i) gegeben durch die beiden Bedingungen

$$\frac{\beta_i}{\beta_0} = \frac{m_i}{n_1 n_2 \cdots n_{i-1} n_i} \quad \text{und} \quad \mathrm{ggT}(m_i, n_i) = 1$$

Schließlich gibt es noch eine Invariante, die sich rekursiv direkt aus der Auflösungskette (1.3) definieren läßt, also ohne Zuhilfenahme von Puiseux-Paaren und damit komplexen Modellen, das *Geschlecht* einer irreduziblen Kurvensingularität.

Definition 1.25 Sei R eine irreduzible ebene algebroide Kurve und R_1 deren quadratische Transformierte. Dann definieren wir das **Geschlecht g** von R folgendermaßen:

- Wenn R regulär ist, sei $g(R) := 0$.
- Wenn $e(R) > e(R_1)$ (die Multiplizitäten) und $e(R_1) \mid e(R)$, dann sei $g(R) := g(R_1) + 1$.
- Sonst $\big(e(R) = e(R_1)$ oder $e(R_1) \nmid e(R)\big)$ sei $g(R) := g(R_1)$.

Bemerkung: Für den Fall, daß man die Puiseux-Paare oder die charakteristischen Exponenten schon kennt, kann man das Geschlecht auch einfach ablesen: Eine Kurve hat genau g Puiseux-Paare bzw. $g+1$ charakteristische Exponenten $\beta_0, \ldots, \beta_g$ (siehe Lemma 4.3 und Satz 4.7 (i)).

1.4 Aufblasen reduzibler algebroider Kurven

Dieser Abschnitt wird gebraucht, um später die Berechnung der Schnittmultiplizität aus den Hamburger-Noether-Entwicklungen der Zweige richtig herleiten zu können.

Die quadratische Transformierte einer irreduziblen algebroiden Kurve ist bereits im letzten Teil (ringtheoretisch) definiert worden. Bei der Verallgemeinerung für den reduziblen Fall habe ich mich an Campillos Darstellung ([Ca, p. 35ff]) orientiert, aber versucht zu vereinfachen. Wir stellen als erstes den Zusammenhang her zwischen der bereits definierten Aufblasung und der quadratischen Transformation, wie man sie normalerweise bei algebraischen Kurven kennenlernt. Zur Vorbereitung definieren wir:

Definition 1.26 Sei $f = \sum_{i \geq n} f_i \in k[\![x,y]\!]$ mit f_i homogen vom Grad i, $f_n \neq 0$. Dann heißt n der **Untergrad** und f_n die (**homogene**) **Leitform** von f.

Bemerkung: Der Untergrad von f ist mit der Multiplizität der entsprechenden Kurve identisch: $n = e(k[\![x,y]\!] \,/\, \langle f \rangle)$ [BK, 5.3].

Folgendes Lemma werden wir auf Seite 45 beweisen:

Lemma 3.1 *Ist $f \in k[\![x,y]\!]$ irreduzibel, dann ist*

$$\underline{\nu}(\mathrm{x}) = \mathrm{ord}_y(f(0,y)) \quad \textit{und} \quad \underline{\nu}(\mathrm{y}) = \mathrm{ord}_x(f(x,0))$$

wobei x *und* y *die Restklassen von x bzw. y in $R = k[\![x,y]\!] \,/\, \langle f \rangle$ sind, $\underline{\nu}$ die zu R assoziierte Bewertung.*

Nun zum Zusammenhang mit der üblichen quadratischen Transformation:

Lemma 1.7 *Sei $R = k[\![x,y]\!] \,/\, \langle f(x,y) \rangle = k[\![\mathrm{x},\mathrm{y}]\!]$ und $n = \underline{\nu}(\mathrm{x}) < \underline{\nu}(\mathrm{y})$. Dann ist*

$$R_1 = k\left[\!\!\left[\mathrm{x}, \frac{\mathrm{y}}{\mathrm{x}}\right]\!\!\right] \cong k[\![x,y]\!] \;/\; \left\langle x^{-n} f(x,xy) \right\rangle$$

Beweis: Nach Lemma 1.3 ist $R_1 = k\,[\![\mathrm{x}, \frac{\mathrm{y}}{\mathrm{x}} - a]\!] = k\,[\![\mathrm{x}, \frac{\mathrm{y}}{\mathrm{x}}]\!]$. Zum Beweis der Isomorphie definieren wir $\varphi : k[\![x,y]\!] \to R_1$ durch $\varphi(x) := \mathrm{x}$, $\varphi(y) := \frac{\mathrm{y}}{\mathrm{x}}$. φ existiert, weil $\underline{\nu}(\mathrm{x}) < \underline{\nu}(\mathrm{y}) \Rightarrow \mathrm{x}, \frac{\mathrm{y}}{\mathrm{x}} \in \mathfrak{m}_1$. φ ist surjektiv; R_1 ist eine ebene algebroide Kurve, daher ist $\operatorname{Ker}\varphi$ ein Hauptideal.

$$\varphi(x^{-n} f(x,xy)) = \mathrm{x}^{-n} \underbrace{f(\mathrm{x}, \mathrm{x}\frac{\mathrm{y}}{\mathrm{x}})}_{=0} = 0 \quad \Longrightarrow \quad x^{-n} f(x,xy) \in \operatorname{Ker}\varphi$$

f ist irreduzibel und nach Lemma 3.1 und Satz 3.2 ist n der Untergrad von $f \Rightarrow x^{-n}f(x,xy)$ ist irreduzibel $\Rightarrow \operatorname{Ker}\varphi = \langle x^{-n} f(x,xy)\rangle$ ▧

Definition 1.27 Wir definieren daher für $f \in k[\![x,y]\!]$ beliebig die **formale** (strikte) **quadratische Transformationsabbildung T** (in Richtung der y-Achse)

$$T : k[\![x,y]\!] \setminus \{0\} \to k[\![x,y]\!] \setminus \{0\} \quad \text{durch} \quad T(f(x,y)) := x^{-n} f(x,xy)$$

wobei n der Untergrad von f ist.

An dieser Stelle zeigt sich ein Unterschied beim Aufblasen zwischen algebraischen und algebroiden Kurven, wir machen nämlich folgende

Beobachtung: *Ist $f = \prod_{i=1}^r f_i \in k[\![x,y]\!]$ reduzibel, $\mathcal{C} := \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f\rangle$ eine algebroide Kurve mit r Zweigen, dann ist $\mathcal{C}' := \operatorname{Spec} k[\![x,y]\!] \,/\, \langle T(f)\rangle$ eine algebroide Kurve mit der Eigenschaft:*

Genau dann ist eine irreduzible algebroide Kurve $\mathcal{D}$ ein Zweig von $\mathcal{C}'$, wenn gilt:

(i) es gibt ein $1 \le i \le r$, so daß $\mathcal{D}$ quadratische Transformierte des Zweiges $\mathcal{C}_i = \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f_i\rangle$ ist **und**

(ii) $\underline{\nu}([x]_{f_i}) < \underline{\nu}([y]_{f_i})$

Beweis: Sei n_i der Untergrad von f_i, dann ist $n = \sum_{i=1}^r n_i$ der Untergrad von f. Mit T wie oben definiert erhalten wir

$$T\left(\prod_{i=1}^r f_i\right) = x^{-n} \prod_{i=1}^r f_i(x,xy) = \prod_{i=1}^r x^{-n_i} f_i(x,xy) = \prod_{i=1}^r T(f_i)$$

$$\Longrightarrow \quad \mathcal{C}' = \operatorname{Spec}\left(k[\![x,y]\!] \,/\, \langle T(f)\rangle\right) = \operatorname{Spec}\left(k[\![x,y]\!] \Big/ \bigcap_{i=1}^r \langle T(f_i)\rangle\right)$$

Wir wählen ein $i \in \{1,\dots,r\}$ und setzen $S := k[\![x,y]\!] \,/\, \langle f_i\rangle$.

Ist $\underline{\nu}([x]_{f_i}) < \underline{\nu}([y]_{f_i})$, so ist nach Lemma 1.7 $S_1 \cong k[\![x,y]\!] \,/\, \langle T(f_i)\rangle$, daher ist $\mathcal{D} := \operatorname{Spec} S_1$ ein Zweig von $\mathcal{C}'$.

Ist jedoch $\underline{\nu}([x]_{f_i}) \ge \underline{\nu}([y]_{f_i}) =: m$, so ist m der Untergrad von f_i, und die Leitform von f_i lautet:

$$\begin{array}{rrcll} & f_{i,m} & = & c_0 x^m + c_1 x^{m-1} y + \ldots + c_m y^m & \text{für } c_i \in k, \quad c_0 \neq 0 \text{ (L. 3.1)} \\ \Longrightarrow & T(f_i) & = & x^{-m} f_{i,m}(x,xy) + x^{-m} \underbrace{(f_i(x,xy) - f_{i,m}(x,xy))}_{\in \langle x^{m+1}\rangle} & \\ & & = & c_0 + c_1 y + \ldots + c_m y^m + Ax & \text{für ein } A \in k[\![x,y]\!] \end{array}$$

Folglich ist $T(f_i)$ in diesem Fall eine Einheit ($c_0 \neq 0$). Wir definieren

$$I := \{1 \leq i \leq r \mid \underline{\nu}([x]_{f_i}) < \underline{\nu}([y]_{f_i})\}, \quad \text{dann erzeugen}$$

$$T(f) = \prod_{i=1}^{r} T(f_i) \quad \text{und} \quad \prod_{i \in I} T(f_i)$$

in $k[\![x,y]\!]$ das gleiche Ideal.

$$\Longrightarrow \quad \mathcal{C}' = \operatorname{Spec}\Big(k[\![x,y]\!] \Big/ \Big\langle \prod_{i\in I} T(f_i)\Big\rangle\Big) = \operatorname{Spec}\Big(k[\![x,y]\!] \Big/ \bigcap_{i\in I} \langle T(f_i)\rangle\Big)$$

Daraus folgt, daß es für jeden Zweig $\mathcal{D}$ von $\mathcal{C}'$ ein $i \in I$ geben muß, so daß $\mathcal{D}$ quadratische Transformierte von $\mathcal{C}_i$ ist, d. h. (i) und (ii) sind erfüllt. ▧

Bemerkung: Die eben gemachte Beobachtung reflektiert einen von algebraischen Kurven her bekannten Sachverhalt: Genau dann schneiden sich die quadratischen Transformierten $\mathcal{C}_i', \mathcal{C}_j'$ zweier Zweige $\mathcal{C}_i, \mathcal{C}_j$ einer Kurve, wenn $\mathcal{C}_i$ und $\mathcal{C}_j$ eine gemeinsame Tangente haben (man beachte, daß für *einen* der beiden Zweige die Bedingung $\underline{\nu}([x]_{f_i}) < \underline{\nu}([y]_{f_i})$ durch Koordinatenwechsel hergestellt werden kann). Allerdings geht bei algebroiden Kurven, wendet man die „klassische" quadratische Transformation an, ein Teil der Zweige verloren, wenn nicht alle die gleiche Tangente haben.

Um dem abzuhelfen, gehen wir folgendermaßen weiter vor:

Definition 1.28 Für eine beliebige ebene algebroide Kurve $\mathcal{C} = \operatorname{Spec} k[\![x,y]\!] / \langle f\rangle$ definiere

$$\boldsymbol{\Delta(\mathcal{C})} := \Delta_f(C) := \left\{ (a:b) \in \mathbb{P}^1_k \;\middle|\; \begin{array}{l} \exists\, n \in \mathbb{N}, \exists\, x(t) = at^n + O(t^{n+1}), \\ \exists\, y(t) = bt^n + O(t^{n+1}) : \\ (x(t), y(t)) \text{ ist Parametrisierung von } f \end{array} \right\}$$

Die Elemente von $\Delta(\mathcal{C})$ nennen wir **Tangentialrichtungen** von $\mathcal{C}$ (man beachte, daß a oder b Null sein kann).

Bemerkung: Ist $\mathcal{C}$ irreduzibel, so besteht $\Delta(\mathcal{C})$ aus genau einem Element, denn ist o.B.d.A. $\underline{\nu}([x]_f) \leq \underline{\nu}([y]_f)$, so ist für eine beliebige Parametrisierung $(x(t), y(t))$ von f

$$\frac{y(t)}{x(t)} = a + z(t) \quad \text{für ein } z(t) \in \langle t\rangle$$

und $a \in k$ ist eindeutig bestimmt durch die Bedingung $\underline{\nu}\left(\frac{[y]_f}{[x]_f} - a\right) > 0$.

Wir verallgemeinern das Lemma 1.7:

Lemma 1.8 (Quadrat. Transformation der definierenden Gleichung) *Sei $R := k[\![x,y]\!] / \langle f\rangle$ der lokale Ring einer irreduziblen algebroiden Kurve $\mathcal{C}$, sei $\Delta(\mathcal{C}) = \{(a:b)\}$ und R_1 die quadratische Transformierte von R. Dann gilt:*

(i) Falls $a \neq 0$, dann ist $R_1 \cong k[\![x,y]\!] \Big/ \Big\langle x^{-e(R)} f\Big(x, xy + \dfrac{b}{a} x\Big)\Big\rangle$

(ii) Falls $a = 0$, dann ist $R_1 \cong k[\![x,y]\!] \Big/ \Big\langle x^{-e(R)} f(xy,x) \Big\rangle$

Beweis: Per definitionem von $\Delta(\mathcal{C})$ gibt es eine (o.B.d.A. primitive) Parametrisierung

$$x(t) = at^n + O(t^{n+1}), \quad y(t) = bt^n + O(t^{n+1}),$$

d. h. $\exists\, u \in \overline{R} : x(u) = [x]_f,\ y(u) = [y]_f$.

(i) $a \neq 0 \;\Rightarrow\; [x]_f$ ist transversal und $\underline{\nu}_R\Big(\frac{[y]_f}{[x]_f} - \frac{b}{a}\Big) = \operatorname{ord}_t\Big(\frac{y(t)}{x(t)} - \frac{b}{a}\Big) > 0$

$\Rightarrow\; \frac{[y]_f}{[x]_f} - \frac{b}{a} \in \mathfrak{m}_R$. Wir machen folgende Transformation:

$$\begin{array}{rcl} S:\ k[\![x,y]\!] \,/\, \langle f(x,y)\rangle & \xrightarrow{\cong} & k[\![x,y]\!] \Big/ \Big\langle f\Big(x, y + \dfrac{b}{a}x\Big)\Big\rangle =: k[\![x,y]\!] \,/\, \langle f'\rangle \\ {[x]}_{f(x,y)} & \mapsto & [x]_{f(x,y+b/a\,x)} \\ {[y]}_{f(x,y)} & \mapsto & \Big[y + \dfrac{b}{a}x\Big]_{f(x,y+b/a\,x)} \end{array}$$

Dann ist $\quad S^{-1}\left(\dfrac{[y]_{f'}}{[x]_{f'}}\right) = \dfrac{\big[y - \frac{b}{a}x\big]_f}{[x]_f} = \dfrac{[y]_f}{[x]_f} - \dfrac{b}{a} \in \mathfrak{m}_R \quad\Longrightarrow\quad \dfrac{[y]_{f'}}{[x]_{f'}} \in \mathfrak{m}_{S(R)}$

$$\Longrightarrow\quad \underline{\nu}([y]_{f'}) > \underline{\nu}([x]_{f'})$$

Mit Lemma 1.7 auf $S(R)$ angewendet erhalten wir

$$(S(R))_1 \cong k[\![x,y]\!] \Big/ \Big\langle x^{-e(S(R))} f'(x,xy)\Big\rangle = k[\![x,y]\!] \Big/ \Big\langle x^{-e(R)} f\Big(x, xy + \frac{b}{a}x\Big)\Big\rangle$$

und wegen $R \cong S(R)$ folgt daraus $R_1 \cong (S(R))_1$, was zu zeigen war.

(ii) $a = 0 \;\Rightarrow\; b \neq 0 \;\Rightarrow\; [y]_f$ ist transversal und $\underline{\nu}([y]_f) < \underline{\nu}([x]_f)$.
Definiere $S:\ k[\![x,y]\!] \,/\, \langle f(x,y)\rangle \xrightarrow{\cong} k[\![x,y]\!] \,/\, \langle f(y,x)\rangle$ und verfahre wie in (i). ▧

Wir definieren daher:

Definition 1.29 Für $(a:b) \in \mathbb{P}^1_k$, $0 \neq f \in k[\![x,y]\!]$ und $n =$ Untergrad von f sei die **formale** (strikte) **quadratische Transformationsabbildung in Richtung** $\boldsymbol{(a:b)}$

$$T_{(a:b)}:\ k[\![x,y]\!] \setminus \{0\} \longrightarrow k[\![x,y]\!] \setminus \{0\}$$

definiert durch

$$T_{(a:b)}(f) := \begin{cases} x^{-n} f\big(x, xy + \frac{b}{a}x\big) & \text{falls } a \neq 0 \\ x^{-n} f(xy,x) & \text{falls } a = 0 \end{cases}$$

Analog zur Beobachtung bei $T = T_{(1:0)}$ gilt für allgemeines T_p, $p \in \mathbb{P}^1_k$:

Lemma 1.9 *Ist $\mathcal{C} = \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f\rangle$ eine algebroide Kurve und $p \in \mathbb{P}^1_k$, ist $f = \prod_{i=1}^r f_i$, f_i irreduzibel und $\mathcal{C}_i = \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f_i\rangle$, dann ist*

$$\langle T_p(f)\rangle = \left\langle \prod_{\substack{i=1 \\ \Delta(\mathcal{C}_i)=\{p\}}}^{r} T_p(f_i) \right\rangle$$

d. h. die Zweige von $\mathcal{C}' := \operatorname{Spec} k[\![x,y]\!] \,/\, \langle T_p(f)\rangle$ *sind genau die quadratischen Transformierten der Zweige* $\mathcal{C}_i$ *von* $\mathcal{C}$ *mit Tangentialrichtung* $\Delta(\mathcal{C}_i) = \{p\}$.

Beweis: Benutze die Transformation S aus dem Beweis des vorausgehenden Lemmas, um die Aussage auf den bereits bewiesenen Fall $p = (1:0)$ zurückzuführen.

Nun sind wir in der Lage, die quadratische Transformierte einer beliebigen (reduziblen) ebenen algebroiden Kurve zu definieren:

Definition 1.30 und Bemerkung: Ist $\mathcal{C} = \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f\rangle$ eine (nicht notwendigerweise irreduzible) algebroide Kurve mit nur einer Tangentialrichtung $\Delta(\mathcal{C}) = \{p\}$, dann definiere die **quadratische Transformierte $\mathcal{C}_1$** von $\mathcal{C}$ durch

$$\mathcal{C}_1 := \operatorname{Spec} k[\![x,y]\!] \,/\, \langle T_p(f)\rangle$$

Diese Definition ist für irreduzibles $\mathcal{C}$ äquivalent zur Definition 1.16, und außerdem sind die Zweige von $\mathcal{C}_1$ genau die quadratischen Transformierten der Zweige von $\mathcal{C}$.

$R_1 := k[\![x,y]\!] \,/\, \langle T_p(f)\rangle$ ist wieder ein lokaler Ring, daher ist $\mathcal{C}_1$ wieder eine algebroide Kurve mit gleicher Anzahl von Zweigen wie $\mathcal{C}$. Wir bezeichnen das maximale Ideal von R_1 auch mit O_1 und nennen $O_1 \in \mathcal{C}_1$ einen **unendlich benachbarten Punkt**.

Ist $\mathcal{C}$ reduzibel mit mehreren Tangentialrichtungen $\Delta(\mathcal{C}) = \{p_1, \ldots, p_s\}$, so verallgemeinern wir

$$\mathcal{C}_1 := \operatorname{Spec}\Big(k[\![x,y]\!] \,/\, \langle T_{p_1}(f)\rangle \oplus \ldots \oplus k[\![x,y]\!] \,/\, \langle T_{p_s}(f)\rangle \Big) =: \operatorname{Spec} R_1$$

In diesem Fall ist $\mathcal{C}_1$ *keine* algebroide Kurve mehr, denn R_1 ist für $s > 1$ kein lokaler Ring! Wir können $\mathcal{C}_1$ aber als disjunkte Vereinigung algebroider Kurven mit je einem unendlich benachbarten Punkt O_{p_i} auffassen: Definiere

$$\mathcal{Z}_i := \operatorname{Spec} k[\![x,y]\!] \,/\, \langle f_i\rangle\,,\ R_{p_i} := k[\![x,y]\!] \Big/ \Big\langle \prod_{\Delta(\mathcal{Z}_j)=\{p_i\}} f_j \Big\rangle \text{ und } \mathcal{C}_{p_i} := \operatorname{Spec} R_{p_i}$$

Dann ist
$$R_1 = (R_{p_1})_1 \oplus \ldots \oplus (R_{p_s})_1$$
und wir schreiben
$$\mathcal{C}_1 = (\mathcal{C}_{p_1})_1 \dot\cup \ldots \dot\cup (\mathcal{C}_{p_s})_1$$

Ist $O_{p_i} \in (\mathcal{C}_{p_i})_1$ das maximale Ideal von $(R_{p_i})_1$, so hat $\mathcal{C}_1$ die unendlich benachbarten Punkte $O_{p_1}, \ldots, O_{p_s}$.

Das Aufblasen läßt sich wie im irreduziblen Fall iterieren: Sei $(R_{p_i})_1$ einer der neu erhaltenen lokalen Ringe, sei $\Delta((\mathcal{C}_{p_i})_1) = \{q_1, \ldots, q_t\}$. Dann erhalten wir bei der quadratischen Transformation von $(R_{p_i})_1$ eine direkte Summe von t neuen lokalen Ringen

$$(R_{p_i})_2 = ((R_{p_i})_{1,q_1})_1 \oplus \ldots \oplus ((R_{p_i})_{1,q_t})_1 = (R_{i1})_2 \oplus \ldots \oplus (R_{it})_2$$

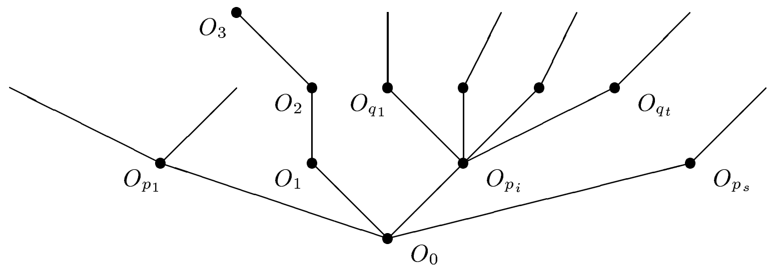
$$\text{wobei} \qquad R_{ij} := k[\![x,y]\!] \Big/ \Big\langle \prod_{\substack{\Delta(Z_l)=\{p_i\} \\ \Delta((Z_l)_1)=\{q_j\}}} f_l \Big\rangle \qquad \text{ist}$$

sowie unendlich benachbarte Punkte $O_{q_1}, \ldots, O_{q_t}$ (die maximalen Ideale von $(R_{p_i})_2$). Zur Unterscheidung der unendlich benachbarten Punkte definieren wir:

Definition 1.31 Ein unendlich benachbarter Punkt von R **in erster Umgebung** ist ein maximales Ideal der quadratischen Transformierten R_1 von R. Ein unendlich benachbarter Punkt von R **in einer Umgebung $(i+1)$-ter Ordnung** ist ein unendlich benachbarter Punkt von R_i (der i-ten quadratischen Transformierten von R) in erster Umgebung.

Eine **Folge unendlich benachbarter Punkte** ist eine Folge $O_0, O_1, \ldots, O_i, \ldots$, so daß O_{i+1} für alle i ein unendlich benachbarter Punkt in erster Umgebung von der Kurve mit abgeschlossenem Punkt O_i ist.

Aus der Auflösungskette wird bei reduziblen Kurven ein Baum:



Dabei entspricht jedem Zweig der Kurve genau ein Weg durch den Baum, und dieser wiederum liefert uns eine Folge unendlich benachbarter Punkte.

Definition 1.32 Sei R eine algebroide Kurve mit r Zweigen. Die **Multiplizitätensequenz** von R ist eine Folge

$$\left[\left(m_i^1, m_i^2, \ldots, m_i^{t_i^1}\right), \left(m_i^{t_i^1+1}, \ldots, m_i^{t_i^2}\right), \ldots, \left(m_i^{t_i^{s_i}+1}, \ldots, m_i^r\right)\right]_{i\in\mathbb{N}}$$

so daß für eine geeignete Sortierung $R^1, \ldots, R^r$ der Zweige von R gilt:

- $m_i^j \in \mathbb{N}$ ist die Multiplizität der i-ten quadratischen Transformierten R_i^j von R^j.
- $R_i^{t_i^l+1}, \ldots, R_i^{t_i^{l+1}}$ schneiden sich im gleichen unendlich benachbarten Punkt von R (in einer Umgebung i-ter Ordnung). Insbesondere: Die Anzahl der unendlich benachbarten Punkte von R in einer Umgebung i-ter Ordnung ist $s_i + 1$.

Bemerkungen:

- Das erste Glied der Multiplizitätensequenz hat immer die Form

$$\left[\left(m_0^1, m_0^2, \ldots, m_0^r\right)\right]$$

(denn die Kurve R selbst besitzt nur einen abgeschlossenen Punkt), und nach endlich vielen Gliedern wird die Folge konstant

$$[\underbrace{(1),(1),\ldots,(1)}_{r \text{ mal}}]$$

(z. B. nach Satz 2.5. Oder explizit: Bemerkung auf S. 130).

- Addiert man die Multiplizitäten $m_i^{t_i^l+1},\ldots,m_i^{t_i^{l+1}}$, so erhält man eine Folge von Multiplizitäten $(M_i^0,\ldots,M_i^{s_i})_{i\in\mathbb{N}}$ mit weniger Information: M_i^j sind die Multiplizitäten der s_i+1 reduziblen Kurven, in die die i-te quadratische Transformierte von R zerfällt (siehe Baum im Bild oben: jede Multiplizität M_i^j gehört zu genau einem Punkt O_{p_j} und wird auch Multiplizität des unendlich benachbarten Punktes O_{p_j} genannt).

2 Hamburger-Noether-Entwicklungen

In diesem Kapitel geht es zunächst einmal darum, Hamburger-Noether-Entwicklungen zu definieren. Wir stellen einen Algorithmus vor, um die Hamburger-Noether-Entwicklung einer irreduziblen Kurvensingularität aus einer Parametrisierung zu berechnen, und betrachten einige wichtige Eigenschaften. Besonders interessiert uns dabei die Frage, wie sich die Entwicklung verändert, wenn man die Singularität aufbläst, und welche Größen in dieser Entwicklung charakteristisch für die Kurve selbst sind, d. h. nicht von der Wahl der Parametrisierung abhängen.

Es sei in diesem Abschnitt stets vorausgesetzt, daß R der lokale Ring einer irreduziblen ebenen algebroiden Kurve $\mathcal{C}$ über dem (beliebigen) Körper k ist und $\overline{R}$ der ganze Abschluß von R im Quotientenkörper $F = Q(R)$ von R. O.B.d.A. gibt es eine quadratfreie Potenzreihe $f \in k[\![X,Y]\!]$, so daß $R = k[\![X,Y]\!] \,/\, \langle f \rangle$ ist (vgl. Bemerkung zu Satz 1.1). Schließlich sei $\mathfrak{m} \subset R$ das maximale Ideal von R und $\underline{\nu}$ die zu R assoziierte Bewertung.

2.1 HNE und Parametrisierung

Definition 2.1 Es sei $\{x, y\}$ ein Erzeugendensystem von $\mathfrak{m}$.
Eine **Hamburger-Noether-Entwicklung** (HNE) von $\{x, y\}$ ist ein Schema

$$\begin{array}{rcl}
z_{-1} & = & a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\
z_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{i-1} & = & a_{i,2}\, z_i^2 + a_{i,3}\, z_i^3 + \cdots\cdots + a_{i,h_i}\, z_i^{h_i} + z_i^{h_i} z_{i+1} \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{r-2} & = & a_{r-1,2}\, z_{r-1}^2 + \cdots + a_{r-1,h_{r-1}}\, z_{r-1}^{h_{r-1}} + z_{r-1}^{h_{r-1}} z_r \\
z_{r-1} & = & a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots\cdots\cdots
\end{array} \tag{HN}$$

wobei $r \geq 0$, $z_{-1}, \ldots, z_r \in \overline{R}$, $1 \leq \underline{\nu}(z_r) < \cdots < \underline{\nu}(z_0) \leq \underline{\nu}(z_{-1})$,[1] $a_{ji} \in k$, $h_i > 0$ und $z_{-1} = y$, $z_0 = x$ oder $z_{-1} = x$, $z_0 = y$ ist.

[1] z_{-1} darf Null sein (alle anderen z_i natürlich nicht); in dem Fall ist $\underline{\nu}(z_{-1})$ aus der Ungleichung wegzulassen.

Man beachte, daß die Summe in der letzten Zeile nicht abbrechen muß; es gibt kein h_r und keinen Summanden $z_r^{h_r} z_{r+1}$.

Eine **Hamburger-Noether-Entwicklung** der algebroiden Kurve R ist eine HNE eines beliebigen Erzeugendensystems $\{x,y\}$ von $\mathfrak{m}$ (auch **Hamburger-Noether-Entwicklung** von R **in der Basis** $\{x,y\}$ genannt). Eine **Hamburger-Noether-Entwicklung von f** ist eine Hamburger-Noether-Entwicklung von $k[\![X,Y]\!] \,/\, \langle f\rangle$ in der Basis $\{[X]_f, [Y]_f\}$.

Der Kürze halber werden wir für (HN) manchmal auch schreiben:

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j}\, z_{j+1}\,, \quad j := 0,\ldots,r \tag{HN$'$}$$

Hier muß man sich aber darüber im klaren sein, daß $a_{j,1} = 0$ für $j > 0$ gilt, und daß die letzte Zeile der HNE nicht von der Form (HN′) ist (man stelle sich $h_r := \infty$ vor).

Die aus den (a_{ji}) bestehende Matrix wird **Hamburger-Noether-Matrix** (HN-Matrix) genannt (diese Matrix hat allerdings keine einheitliche Spaltenzahl, oder man setzt die noch undefinierten Einträge auf $a_{ji} := 0$ für $i > h_j$). Die Hamburger-Noether-Matrix spielt erst in Teil II eine Rolle – dort dient sie als Struktur zur Speicherung der algorithmisch berechneten HNE und wird zu diesem Zweck in ihrer Definition noch etwas modifiziert.

Die Koeffizienten a_{ji} und die Exponenten h_j von (HN) sind die interessanten Bestandteile der HNE – um ihre Berechnung geht es in der vorliegenden Arbeit. Wir nennen zwei Hamburger-Noether-Entwicklungen **gleich**, wenn sie in den Koeffizienten und Exponenten (und natürlich auch in der Zeilenanzahl) übereinstimmen.

Aus einer Hamburger-Noether-Entwicklung läßt sich sofort eine Parametrisierung der betreffenden Kurve berechnen:

Lemma 2.1 *Substituiere in der letzten Zeile von* (HN) $z_r := T$ *(T eine Unbestimmte). Dann erhalten wir eine Potenzreihe* $z_{r-1}(T) \in k[\![T]\!]$. *Rückwärtseinsetzen liefert Potenzreihen* $z_{r-1}(T), \ldots, z_{-1}(T) \in k[\![T]\!]$. *Definiere* $\mathrm{x}(T) := z_0(T)$, $\mathrm{y}(T) := z_{-1}(T)$ *falls* $x = z_0$, $y = z_{-1}$ *ist, sonst vertausche* $\mathrm{x}(T), \mathrm{y}(T)$.

Dann ist $(\mathrm{x}(T), \mathrm{y}(T))$ *eine primitive lokale parametrische Darstellung von R in der Basis* (x,y).

Beweis: Per definitionem gilt $x = \mathrm{x}(z_r)$, $y = \mathrm{y}(z_r)$, daher ist $(\mathrm{x}(T), \mathrm{y}(T))$ eine Parametrisierung, und weil $z_r \in \overline{R} \subset Q(R)$ ist, ist die Parametrisierung primitiv (vergleiche Definition 1.7 auf S. 12). ▧

Korollar 1 *Bei der HNE von* $\{x,y\}$ *ist z_r ein regulärer Parameter, insbesondere ist also* $\underline{\nu}(z_r) = 1$.

Beweis: $z_r \in \overline{R} \Rightarrow k[\![z_r]\!] \subseteq \overline{R}$; $R = k[\![x,y]\!] = k[\![\mathrm{x}(z_r), \mathrm{y}(z_r)]\!] \subseteq k[\![z_r]\!]$, $k[\![z_r]\!]$ ist ganz abgeschlossen $\Rightarrow \overline{R} \subseteq k[\![z_r]\!] \Rightarrow k[\![z_r]\!] = \overline{R}$. ▧

Korollar 2 *Für jedes Schema* (HN) *einer HNE gibt es eine algebroide Kurve R, so daß* (HN) *eine Hamburger-Noether-Entwicklung von R ist.*

Beweis: Substituiere wie in Lemma 2.1 z_r durch eine Unbestimmte T und erhalte zwei Potenzreihen $x(T), y(T)$. Definiere $R := k[\![x(T), y(T)]\!] \subset k[\![T]\!]$. Dann erzeugen $x(T), y(T)$ das maximale Ideal von R und (HN) ist eine HNE von $\{x(T), y(T)\}$. ▧

Beispiel: Sei $R := k[\![t^2, t^3]\!] \subset k[\![t]\!]$ und $x := t^2$, $y := t^3$ als Erzeugendensystem von $\mathfrak{m}$ gewählt. Wir setzen

$$z_{-1} := y, \qquad z_0 := x, \qquad z_1 := t$$

Dann gilt:

$$\begin{aligned} z_{-1} &= z_0 \, z_1 \\ z_0 &= z_1^2 \end{aligned}$$

Dies ist eine Hamburger-Noether-Entwicklung von $\{x, y\}$, es gilt nämlich: $r = 1$, $h_0 = 1$, $a_{0,1} = 0$, $a_{1,2} = 1$ und $a_{1,i} = 0$ für alle $i > 2$.
Die Parametrisierung, die wir hiervon ablesen können, lautet (setze $z_1 := T$):

$$\mathrm{x}(T) = z_0(T) = T^2 \qquad \mathrm{y}(T) = z_0(T) z_1(T) = T^2 T = T^3$$

Unser erstes Ziel ist es zu zeigen, daß es für jedes Erzeugendensystem $\{x, y\}$ von $\mathfrak{m}$ *genau eine* (in Ausnahmefällen: zwei) Hamburger-Noether-Entwicklung(en) gibt. Die Eindeutigkeit beruht auf folgendem

Satz 2.2 *Ist $\{x, y\}$ Erzeugendensystem von $\mathfrak{m}$ und x transversaler Parameter, so ist die Hamburger-Noether-Entwicklung von $\{x, y\}$ eindeutig durch die Forderung $z_0 = x$, $z_{-1} = y$ bestimmt.*

Beweis: Für (HN) eine HNE von $\{x, y\}$ wie auf Seite 25 definieren wir

$$L(\mathsf{HN}) := h_0 + h_1 + \ldots + h_{r-1} \qquad \text{und}$$

$$\mathcal{L}(\{x, y\}) := \min \big\{ L(\mathsf{HN}) \;\big|\; (\mathsf{HN}) \text{ ist HNE für } \{x, y\} \big\}$$

Wir benutzen Induktion über $\mathcal{L}(\{x, y\})$:
I.A.: Sei $\mathcal{L}(\{x, y\}) = 0$. Dann gibt es eine HNE (HN) von $\{x, y\}$ mit $r = 0$, da ja bereits $h_0 > 0$ wäre, wenn es existieren würde. Das heißt also

$$\begin{aligned} (\mathsf{HN}) \qquad z_{-1} &= a_{0,1} \, z_0 + a_{0,2} \, z_0^2 + \cdots \qquad \text{ist eine HNE für } \{x, y\} \\ \Longrightarrow \qquad y &= a_{0,1} \, x + a_{0,2} \, x^2 + \cdots \end{aligned}$$

Sei nun (HN') eine beliebige HNE für $\{x, y\}$. Dann gilt einer der beiden Fälle:

(A) $y = a'_{0,1} x + a'_{0,2} x^2 + \cdots + a'_{0,h'_0} x^{h'_0} + x^{h'_0} z'_1$ für ein $z'_1 \in \overline{R}$
mit $1 \leq \underline{\nu}(z_1) < \underline{\nu}(x)$ (falls (HN') mehrere Zeilen hat), *oder*

(B) $y = a'_{0,1} x + a'_{0,2} x^2 + \cdots$ (falls (HN') auch nur aus einer Zeile besteht).

Angenommen es gilt (A). Dann würde folgen:

$$\underbrace{\sum_{i=1}^{h_0'} \left(a_{0i}' - a_{0i}\right) x^i}_{=0 \text{ oder } \underline{\nu}() \leq h_0' \cdot \underline{\nu}(x)} = \underbrace{\left(\sum_{i>h_0'} a_{0i}\, x^i\right) - x^{h_0'} z_1'}_{=0 \text{ oder } \underline{\nu}() > h_0' \cdot \underline{\nu}(x)}$$

$$\begin{aligned}
\Longrightarrow\quad & \sum_{i=1}^{h_0'} \left(a_{0i}' - a_{0i}\right) x^i = 0 = \left(\sum_{i>h_0'} a_{0i}\, x^i\right) - x^{h_0'} z_1' \\
\Longrightarrow\quad & \sum_{i>h_0'} a_{0i} x^i = x^{h_0'} z_1' \neq 0 \\
\Longrightarrow\quad & \underline{\nu}(x) \;\Big|\; \underline{\nu}\left(\sum_{i>h_0'} a_{0i}\, x^i\right) = \underline{\nu}\left(x^{h_0'} z_1'\right) = \underline{\nu}(x) h_0' + \underline{\nu}(z_1') \\
\Longrightarrow\quad & \underline{\nu}(x) \;\big|\; \underline{\nu}(z_1'), \quad \text{im Widerspruch zu} \quad 1 \leq \underline{\nu}(z_1') < \underline{\nu}(x) \quad \lightning
\end{aligned}$$

Also muß Fall (B) gelten. Es folgt (weil $\overline{R}$ ein diskreter Bewertungsring ist):

$$\sum_{i\in\mathbb{N}} (a_{0i} - a_{0i}')x^i = 0 \;\Rightarrow\; \forall\, i \in \mathbb{N} : a_{0i} = a_{0i}' \;\Rightarrow\; (\mathsf{HN}) = (\mathsf{HN}')$$

I.V.: Sei $N > 0$ und die Behauptung richtig für alle $\{x', y'\}$ mit $\mathcal{L}(\{x', y'\}) < N$.
I.S.: Wir betrachten ein Erzeugendensystem $\{x, y\}$ mit $\mathcal{L}(\{x, y\}) = N$. Dieses Erzeugendensystem besitzt also eine HNE (HN), für die $L(\mathsf{HN}) = N$ ist:

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji} z_j^i + z_j^{h_j} z_{j+1}\,, \quad j := 0, \ldots, r \tag{HN}$$

Sei (HN′) eine weitere HNE von $\{x, y\}$:

$$z_{j-1}' = \sum_{i=1}^{h_j'} a_{ji}' {z_j'}^i + {z_j'}^{h_j'} z_{j+1}'\,, \quad j := 0, \ldots, r' \tag{HN′}$$

Wegen $L(\mathsf{HN}') \geq L(\mathsf{HN}) = N > 0$ gilt $r > 0$, $r' > 0$. Da nach Voraussetzung $z_0 = x = z_0'$, $z_{-1} = y = z_{-1}'$ ist, erhalten wir

$$a_{01}x + a_{02}x^2 + \cdots + a_{0h_0}x^{h_0} + x^{h_0} z_1 = y = a_{01}'x + a_{02}'x^2 + \cdots + a_{0h_0'}' x^{h_0'} + x^{h_0'} z_1'$$

aus der jeweils ersten Zeile von (HN) und (HN′). Definiere $y_1 := (y - a_{01}x)/x$.

$$\begin{aligned}
\Longrightarrow\; y_1 \;&=\; a_{02}x + \cdots + a_{0h_0}x^{h_0-1} + x^{h_0-1} z_1 \\
&=\; (a_{01}' - a_{01}) + a_{02}'x + \cdots + a_{0h_0'}' x^{h_0'-1} + x^{h_0'-1} z_1'
\end{aligned}$$

Fall 1: $h_0 > 1$. Aus der ersten Gleichung für y_1 ergibt sich, daß $\underline{\nu}(y_1) \geq \underline{\nu}(x) \geq 1$ ist, daher ist $a_{01}' - a_{01} = 0$ und wir können eine neue Kurve betrachten, deren maximales Ideal $\{x, y_1\}$ als Erzeugendensystem hat; auch für diese Kurve ist x ein transversaler Parameter.

Ausgehend von (HN) und (HN′) können wir sofort zwei Hamburger-Noether-Entwicklungen für $\{x, y_1\}$ angeben:

$$\begin{array}{rcl} y_1 = z_{-1} & = & a_{0,2}\, z_0 + a_{0,3}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0-1} + z_0^{h_0-1} z_1 \\ x = z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\ z_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ z_{r-1} & = & a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots\cdots\cdots\cdots \end{array} \qquad (\mathsf{HN}_1)$$

$$\begin{array}{rcl} y_1 = z'_{-1} & = & a'_{0,2}\, z'_0 + a'_{0,3}\, {z'_0}^2 + \cdots\cdots + a'_{0,h'_0}\, {z'_0}^{h'_0-1} + {z'_0}^{h'_0-1} z'_1 \\ x = z'_0 & = & a'_{1,2}\, {z'_1}^2 + a'_{1,3}\, {z'_1}^3 + \cdots\cdots + a'_{1,h'_1}\, {z'_1}^{h'_1} + {z'_1}^{h'_1} z'_2 \\ z'_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ z'_{r'-1} & = & a'_{r',2}\, {z'_{r'}}^2 + a'_{r',3}\, {z'_{r'}}^3 + \cdots\cdots\cdots\cdots\cdots\cdots \end{array} \qquad (\mathsf{HN}'_1)$$

Es gilt $L(\mathsf{HN}_1) = h_0 - 1 + h_1 + \cdots + h_{r-1} = N - 1$, daher ist $\mathcal{L}(\{x, y_1\}) \le N - 1$, also erfüllt $\{x, y_1\}$ die Induktionsvoraussetzung. Daraus folgt $(\mathsf{HN}_1) = (\mathsf{HN}'_1)$, d. h.:

$$r = r', \qquad h_i = h'_i \quad \forall\, i := 0, \ldots, r-1\,, \qquad a_{ij} = a'_{ij} \quad \forall\, 2 \le j \le h_i$$

Wir haben aber schon gezeigt, daß auch $a_{01} = a'_{01}$ ist. Folglich ist (HN) = (HN′).

Fall 2: $h_0 = 1$. Dann ist $y_1 = z_1 \;\Rightarrow\; \underline{\nu}(y_1) = \underline{\nu}(z_1) \ge 1$, und wiederum ist $a'_{01} = a_{01}$. Wäre $h'_0 > 1$, so folgte aus der zweiten Gleichung für y_1, daß $\underline{\nu}(x) \le \underline{\nu}(y_1) = \underline{\nu}(z_1)$ ist, was nicht sein kann. Also gilt auch $y_1 = z'_1$, und wir kennen zwei Hamburger-Noether-Entwicklungen von $\{x, y_1\}$, nämlich

$$\begin{array}{rcl} x = z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\ y_1 = z_1 & = & a_{2,2}\, z_2^2 + a_{2,3}\, z_2^3 + \cdots\cdots + a_{2,h_2}\, z_2^{h_2} + z_2^{h_2} z_3 \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{array} \qquad (\mathsf{HN}_1)$$

$$\begin{array}{rcl} x = z'_0 & = & a'_{1,2}\, {z'_1}^2 + a'_{1,3}\, {z'_1}^3 + \cdots\cdots + a'_{1,h'_1}\, {z'_1}^{h'_1} + {z'_1}^{h'_1} z'_2 \\ y_1 = z'_1 & = & a'_{2,2}\, {z'_2}^2 + a'_{2,3}\, {z'_2}^3 + \cdots\cdots + a'_{2,h'_2}\, {z'_2}^{h'_2} + {z'_2}^{h'_2} z'_3 \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{array} \qquad (\mathsf{HN}'_1)$$

Wie in Fall 1 erfüllt $\{x, y_1\}$ die Induktionsvoraussetzung wegen

$$L(\mathsf{HN}_1) = h_1 + \ldots + h_{r-1} = N - 1\,,$$

woraus $(\mathsf{HN}_1) = (\mathsf{HN}'_1)$ und schließlich (HN) = (HN′) folgt. ▧

2.2 Der Algorithmus

Nun kommen wir zur Frage nach der Existenz einer Hamburger-Noether-Entwicklung. Dazu geben wir einen Algorithmus an, der aus einer vorgegebenen Parametrisierung eine Hamburger-Noether-Entwicklung berechnet. Damit ist klar: Jede algebroide Kurve, die sich parametrisieren läßt, besitzt auch eine Hamburger-Noether-Entwicklung; dies gilt insbesondere für alle Kurven über algebraisch abgeschlossenen Körpern.

Algorithmus `Param-HNE`

Voraussetzung: $\{\mathrm{x},\mathrm{y}\}$ sei Erzeugendensystem des maximalen Ideals $\mathfrak{m}$ von R, x sei transversal.

Eingabe: $x, y \in k[[T]]$, so daß (x, y) lokale parametrische Darstellung von R in der Basis (x,y) ist.

Ausgabe: Die Koeffizienten a_{ji} und die Exponenten h_i der Hamburger-Noether-Entwicklung von $\{\mathrm{x},\mathrm{y}\}$

Schritt 0: $z_{-1} \leftarrow y, \quad z_0 \leftarrow x, \quad s \leftarrow 0$

Schritt 1: $i \leftarrow 1, \quad y_0 \leftarrow z_{s-1}$

Schritt 2:

Berechne $a_{si} \in k$ und $y_i \in k[[T]]$ mit $\frac{y_{i-1}}{z_s} = a_{si} + y_i$ und $\mathrm{ord}_T(y_i) > 0$.
`falls` $\mathrm{ord}(y_i) \geq \mathrm{ord}(z_s)$
`dann` setze $i \leftarrow i+1$ und wiederhole Schritt 2
`sonst` setze $h_s \leftarrow i$ (und weiter mit Schritt 3)

Schritt 3: $s \leftarrow s+1, \quad z_s \leftarrow y_{h_{s-1}},$ gehe zurück nach Schritt 1.

Bemerkung: Dieser Algorithmus taugt natürlich noch nicht zur praktischen Berechnung, denn er terminiert nicht. Das kann auch gar nicht sein, denn in der letzten Zeile einer HNE gibt es im allgemeinen unendlich viele Koeffizienten a_{ri} – nur in Ausnahmefällen sind davon alle bis auf endlich viele gleich Null.

Der Algorithmus leistet das gewünschte; Beweis: Per definitionem gibt es eine endliche Körpererweiterung $L \supseteq Q(R)$ und ein $u \in L$: $\mathrm{x} = x(u)$, $\mathrm{y} = y(u)$ (vgl. (1.2), S. 12). Aus Schritt 0 ergibt sich folglich $z_{-1}(u) = \mathrm{y}$, $z_0(u) = \mathrm{x}$. Folgende Beobachtungen können wir machen:

(i) Jedesmal, wenn Schritt 3 erreicht wird, also nach der Anweisung $h_s \leftarrow i$, gilt:

$$\forall\, 1 \leq i \leq h_s : \qquad \frac{y_{i-1}}{z_s} = a_{si} + y_i \quad \Leftrightarrow \quad y_{i-1} = a_{si}\, z_s + y_i\, z_s$$

$$\begin{aligned} \Longrightarrow \quad z_{s-1} &\underset{(1)}{=} y_0 = a_{s1}\, z_s + y_1\, z_s = a_{s1}\, z_s + (a_{s2}\, z_s + y_2\, z_s) z_s = \dots \\ &= a_{s1}\, z_s + a_{s2}\, z_s^2 + \ldots + a_{sh_s}\, z_s^{h_s} + y_{h_s}\, z_s^{h_s} \\ \underset{(3)}{\Longrightarrow} \quad z_{s-1} &= a_{s1}\, z_s + a_{s2}\, z_s^2 + \ldots + a_{sh_s}\, z_s^{h_s} + z_s^{h_s}\, z_{s+1} \end{aligned} \tag{2.1}$$

(ii) Wegen der Bedingung für den `sonst`-Zweig ist $\mathrm{ord}(z_{s+1}) < ord(z_s)$.

Aus diesem Grund wird Schritt 3 nur endlich oft erreicht, denn wegen der Definition von y_i in Schritt 2 gilt

$$\mathrm{ord}(z_{-1}) \geq \mathrm{ord}(z_0) > \mathrm{ord}(z_1) > \ldots > \mathrm{ord}(z_s) > \ldots \geq 1$$

Das heißt: Es werden nur *endlich* viele $z_{-1}, z_0, \ldots, z_{r-1}$ berechnet mit der Eigenschaft (2.1).

(iii) Nach endlich vielen Schritten wird also nur noch Schritt 2 ausgeführt; in diesem Fall gilt:

$$\forall\, 1 \leq i < \infty: \qquad \frac{y_{i-1}}{z_s} = a_{si} + y_i \quad \Leftrightarrow \quad y_{i-1} = a_{si}\, z_s + y_i\, z_s$$

$$\begin{aligned} \Longrightarrow \quad z_{s-1} &\underset{(1)}{=} y_0 = a_{s1}\, z_s + y_1\, z_s = a_{s1}\, z_s + (a_{s2}\, z_s + y_2\, z_s) z_s = \ldots \\ &= a_{s1}\, z_s + a_{s2}\, z_s^2 + a_{s3}\, z_s^3 + \ldots \end{aligned} \tag{2.2}$$

(iv) Betrachte den Fall $s > 0$ und $i = 1$ in Schritt 2: Nach (ii) gilt $\mathrm{ord}(z_s) < \mathrm{ord}(z_{s-1})$, daher ist

$$\mathrm{ord}\left(\frac{y_0}{z_s}\right) = \mathrm{ord}\left(\frac{z_{s-1}}{z_s}\right) \geq 1$$

Wir erhalten: $\forall\, s > 0 \,:\, a_{s1} = 0$

(v) Trivialerweise gilt auch $h_s \geq 1$ für alle s.

Ergebnis: Für die im Verlauf des Algorithmus berechneten Potenzreihen z_s gilt:

$$\begin{aligned} y = z_{-1} &= a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\ x = z_0 &= a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\ z_1 &= \cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots &= \cdots\cdots\cdots\cdots\cdots\cdots \\ z_{s-1} &= a_{s,2}\, z_s^2 + a_{s,3}\, z_s^3 + \cdots\cdots + a_{s,h_s}\, z_s^{h_s} + z_s^{h_s} z_{s+1} \\ \cdots &= \cdots\cdots\cdots\cdots\cdots\cdots \\ z_{r-1} &= a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots \end{aligned} \tag{2.3}$$

In alle diese Potenzreihen setzen wir nun u ein und erhalten folgendes Schema:

$$\begin{aligned} \mathrm{y} = z_{-1}(u) &= a_{0,1}\, z_0(u) + a_{0,2}\, z_0(u)^2 + \cdots\cdots + a_{0,h_0}\, z_0(u)^{h_0} + z_0(u)^{h_0} z_1(u) \\ \mathrm{x} = z_0(u) &= a_{1,2}\, z_1(u)^2 + a_{1,3}\, z_1(u)^3 + \cdots\cdots + a_{1,h_1}\, z_1(u)^{h_1} + z_1(u)^{h_1} z_2(u) \\ z_1(u) &= \cdots\cdots\cdots\cdots\cdots\cdots \\ \cdots &= \cdots\cdots\cdots\cdots\cdots\cdots \\ z_{r-1}(u) &= a_{r,2}\, z_r(u)^2 + a_{r,3}\, z_r(u)^3 + \cdots\cdots\cdots \end{aligned} \tag{2.4}$$

wobei $a_{ji} \in k$ und $h_i > 0$. Wir müssen nun zeigen, daß die $z_s(u)$ überhaupt Elemente von $\overline{R}$ sind – zunächst ist ja nur klar, daß $z_s(u) \in L \;\; \forall s$ – und daß $\underline{\nu}(z_{s-1}(u)) > \underline{\nu}(z_s(u)) \;\; \forall\, s > 0$ ist.

Sei dazu $t \in \overline{R}$ ein regulärer Parameter, dann gibt es eine Potenzreihe $p \in k[\![T]\!]$: $p(u) = t$, denn $\mathrm{x}, \mathrm{y} \in k[\![u]\!] \;\Rightarrow\; R = k[\![\mathrm{x}, \mathrm{y}]\!] \subset k[\![u]\!] \;\Rightarrow\; \overline{R} \subset \overline{k[\![u]\!]} = k[\![u]\!]$.

Induktionsanfang: $z_{-1}(u) = \mathrm{y}, z_0(u) = \mathrm{x} \in \mathfrak{m} \subset \overline{R}, \;\; \underline{\nu}(z_{-1}(u)) \geq \underline{\nu}(z_0(u))$
I.V.: Sei $0 \leq s < r, \;\; z_{s-1}(u), z_s(u) \in \overline{R}$ und $\underline{\nu}(z_{s-1}(u)) \geq \underline{\nu}(z_s(u))$.

I.S.: $$z_s(u)^{h_s} z_{s+1}(u) = z_{s-1}(u) - \sum_{i=1}^{h_s} a_{si}\, z_s(u)^i \in \overline{R} \qquad \text{nach } I.V., (2.4)$$

$$\Longrightarrow \quad z_{s+1}(u) \in Q(\overline{R}) = Q(R)$$

Daß $z_{s+1}(u)$ sogar in $\overline{R}$ liegt, zeigen wir durch Konstruktion einer Potenzreihe $P : z_{s+1}(u) = P(t)$. Dann folgt die Behauptung nämlich aus $t \in \overline{R}$.

Nach I.V. gibt es Potenzreihen $z'_{s-1}, z'_s \in k[\![T]\!] : z'_{s-1}(t) = z_{s-1}(u),\; z'_s(t) = z_s(u)$.

$$\Longrightarrow \quad z'_i(p(u)) = z_i(u) \quad \Longrightarrow \quad z'_i(p) = z_i \qquad \text{für } i = s-1,\; s$$
$$\Longrightarrow \quad z'_{s-1}(p) = \sum_{i=1}^{h_s} a_{si}\, {z'_s}^i(p) + {z'_s}^{h_s}(p)\, z_{s+1} \qquad \text{nach (2.1)}$$

Wir betrachten die Schritte 1 & 2 des Algorithmus für z_s und z'_s:

Sei $y'_0 := z'_{s-1}$ und $1 \leq i \leq h_s$ so gewählt, daß $y'_{i-1}(p) = y_{i-1}$. Definiere a'_{si}, y'_i analog zu a_{si}, y_i durch $\frac{y'_{i-1}}{z'_s} = a'_{si} + y'_i$. Dann folgt:

$$a_{si} + y_i = \frac{y_{i-1}}{z_s} = \frac{y'_{i-1}(p)}{z'_s(p)} = a'_{si} + y'_i(p) \quad \Longrightarrow \quad a_{si} = a'_{si},\; y_i = y'_i(p)$$
$$i \leq h_s \;\Rightarrow\; \mathrm{ord}(y_{i-1}) \geq \mathrm{ord}(z_s) \;\Rightarrow\; \mathrm{ord}(y'_{i-1}) \geq \mathrm{ord}(z'_s) \quad \Longrightarrow \quad \frac{y'_{i-1}}{z'_s} \in k[\![T]\!]$$
$$\Longrightarrow \quad y'_i \in k[\![T]\!].$$

Mit Induktion folgt: $\exists\, y'_{h_s} \in k[\![T]\!] : \; z'_{s-1} = \sum_{i=1}^{h_s} a_{si}\, {z'_s}^i + {z'_s}^{h_s} y'_{h_s}$

$$\Longrightarrow \quad y'_{h_s}(p) = \frac{z'_{s-1}(p) - \sum_{i=1}^{h_s} a_{si}\, z'_s(p)^i}{z'_s(p)^{h_s}} = \frac{z_{s-1} - \sum_{i=1}^{h_s} a_{si}\, z_s^i}{z_s^{h_s}} = z_{s+1}$$
$$\Longrightarrow \quad z_{s+1}(u) = y'_{h_s}(p(u)) = y'_{h_s}(t) \in \overline{R}$$

Schließlich folgt aus $\mathrm{ord}(z_s) > \mathrm{ord}(z_{s+1})$, daß auch $\underline{\nu}(z_s(u)) > \underline{\nu}(z_{s+1}(u))$ erfüllt ist.

Damit haben wir gezeigt, daß (2.4) eine Hamburger-Noether-Entwicklung von $\{\mathrm{x}, \mathrm{y}\}$ ist. ▨

Fazit: Sei k algebraisch abgeschlossen und $\{\mathrm{x},\mathrm{y}\}$ Erzeugendensystem des maximalen Ideals von R. Ist nur einer der beiden Parameter transversal (d. h. $\underline{\nu}(\mathrm{x}) \neq \underline{\nu}(\mathrm{y})$), dann gibt es *genau eine* Hamburger-Noether-Entwicklung von $\{\mathrm{x},\mathrm{y}\}$. Sind beide transversal, so gibt es *genau zwei* HNEs von $\{\mathrm{x},\mathrm{y}\}$, nämlich eine mit $z_{-1} = \mathrm{y}$, $z_0 = \mathrm{x}$, und eine mit $z_{-1} = \mathrm{x}$, $z_0 = \mathrm{y}$.

Definition 2.2 Ein Schema von Potenzreihen wie in (2.3) heißt **Hamburger-Noether-Entwicklung der Parametrisierung** (x, y).

Korollar aus Satz 2.2 *Die Ausgabe des Algorithmus hängt nicht von der Wahl der parametrischen Darstellung ab, d. h.*

$$\textit{wenn} \quad x', y' \in k[[T]] \quad \textit{sind, so daß} \quad \exists\, u' \in L' \supseteq Q(R) : x'(u') = \mathrm{x},\ y'(u') = \mathrm{y},$$

dann liefert der Algorithmus für (x', y') dasselbe wie für (x, y). Die Hamburger-Noether-Entwicklungen von (x, y) und (x', y') sind also gleich im Sinne der Definition auf Seite 26.

Außerdem ist die Hamburger-Noether-Entwicklung von R in der Basis $\{\mathrm{x},\mathrm{y}\}$ *gleich der HNE einer beliebigen lokalen parametrischen Darstellung von R in der Basis* $\{\mathrm{x},\mathrm{y}\}$. *(Insbesondere ist auch die HNE einer* Parametrisierung *eindeutig.)*

Nun kommen wir zur Frage nach der praktischen Berechnung. Der Algorithmus `Param-HNE` erlaubt im Prinzip eine beliebig genaue Berechnung der Hamburger-Noether-Entwicklung, aber will man eine HNE konkret berechnen, muß man irgendwann die Rechnung abbrechen, es sei denn im Schritt 2 wird irgendwann ein y_i „zufällig“ Null – dann ist klar, daß $y_j = 0\ \forall\, j \geq i$ und damit $a_{sj} = 0$ $\forall\, j \geq i$, d. h. die HNE ist endlich. Es ist auch nicht so, daß für den Fall, in dem $x, y \in k[T]$ Polynome sind (normalerweise werden nur solche Parametrisierungen gegeben sein, insbesondere bei der Implementation auf dem Computer), die HNE von (x, y) automatisch endlich ist.

Beispiel: Wähle $x := T^2 + T^3$, $y := T^3$. Wir wissen schon, daß die HNE von (x,y) eindeutig ist, und daß der Algorithmus `Param-HNE` sie liefert. Wir benutzen also diesen Algorithmus, um den Anfang der HNE auszurechnen:

Schritt 0: $z_{-1} := y$, $z_0 := x$; Schritt 1: $y_0 := z_{-1}$.
Schritt 2: $\frac{y_0}{z_0} = \frac{y}{x} = \frac{T}{1+T} =: y_1,\ a_{01} := 0.$ $\mathrm{ord}(y_1) = 1 < \mathrm{ord}(z_0)$, daher setze $h_0 := 1$ und gehe zu Schritt 3: $s := 1$, $z_1 := y_1$. Schritt 1: $y_0 := z_0$
Schritt 2: $\frac{y_0}{z_1} = \frac{(T^2+T^3)(1+T)}{T} = T(1+T)^2 =: y_1,\ a_{11} := 0.$ $\frac{y_1}{z_1} = (1+T)^3$, $a_{12} := 1$. Da $\mathrm{ord}(z_1) = 1$ gilt, ist die Bedingung $\mathrm{ord}(y_i) \geq \mathrm{ord}(z_1)$ im folgenden immer erfüllt, d. h. Schritt 3 wird nicht mehr durchlaufen; wir sind in der letzten Zeile der HNE und brechen die Berechnung ab.

Die bis jetzt berechnete HNE lautet also $\begin{array}{l} y = z_{-1} = z_0 z_1 \\ x = z_0 = z_1^2 + \ldots \end{array}$

Angenommen die HNE wäre endlich, d. h. $\exists\, a_2, \ldots, a_h \in k : x = a_2 z_1^2 + \ldots + a_h z_1^h$

$$\Longrightarrow \quad T^2 + T^3 \;=\; a_2 \left(\frac{T}{1+T}\right)^2 + a_3 \left(\frac{T}{1+T}\right)^3 + \ldots + a_h \left(\frac{T}{1+T}\right)^h$$

$$
\begin{aligned}
&= \frac{T^2}{(1+T)^h}\Big(a_2(1+T)^{h-2} + a_3T(1+T)^{h-3} + \ldots + a_hT^{h-2}\Big)\\
\Longrightarrow\ (1+T)^{h+1} &= a_2(1+T)^{h-2} + a_3T(1+T)^{h-3} + \ldots + a_hT^{h-2} \quad \lightning\\
&\ \text{(verschiedene Grade links und rechts)}
\end{aligned}
$$

Die vollständige HNE lautet folglich
$$
\begin{aligned}
y &= z_{-1} = z_0z_1\\
x &= z_0 = \textstyle\sum_{i=2}^{\infty} a_i z_1^i
\end{aligned}
$$

wobei unendlich viele der a_i nicht Null sind. Obwohl also x und y Polynome sind, bricht die HNE von (x, y) nicht ab.

Wie bekommen wir also einen Algorithmus, der mit Sicherheit terminiert? Es ist sinnvoll, die Berechnung der HNE abzubrechen, wenn die letzte Zeile erreicht ist, denn alle anderen Zeilen besitzen nur endlich viele Summanden und sind daher in endlich vielen Schritten berechenbar. Es wird sich außerdem später herausstellen, daß man ein vollständiges System von Invarianten besitzt, sobald man so weit gerechnet hat, daß man einen nichtverschwindenden Koeffizienten in der letzten Zeile kennt. Es bleibt die Frage, woher man weiß, daß die letzte Zeile der HNE erreicht ist. Hierbei hilft uns folgendes

Lemma 2.3 *Genau dann ist $\{x, y\} \subset k\llbracket T\rrbracket$ eine* primitive *Parametrisierung (irgendeiner Kurve), wenn für z_r aus der HNE von (x, y) gilt (gleiche Bezeichnung wie in Schema 2.3):* $\operatorname{ord}(z_r) = 1$.

Beweis: Wir setzen voraus, daß $\{x, y\}$ eine primitive Parametrisierung ist.

$$
\begin{aligned}
\Longrightarrow\quad & \exists\, u \in Q(R) : x(u) = \mathrm{x},\ y(u) = \mathrm{y},\ R = k\llbracket \mathrm{x}, \mathrm{y}\rrbracket = k\llbracket x(u), y(u)\rrbracket\\
& z_r \text{ ist so konstruiert, daß } x, y \in k\llbracket z_r\rrbracket \text{ sind.} \implies \mathrm{x}, \mathrm{y} \in k\llbracket z_r(u)\rrbracket\\
\Longrightarrow\quad & R = k\llbracket \mathrm{x}, \mathrm{y}\rrbracket \subseteq k\llbracket z_r(u)\rrbracket \subseteq k\llbracket u\rrbracket \subseteq \overline{R}\\
& k\llbracket z_r(u)\rrbracket \text{ und } k\llbracket u\rrbracket \text{ sind ganz abgeschlossen}\\
\Longrightarrow\quad & \overline{R} \subseteq k\llbracket z_r(u)\rrbracket \subseteq k\llbracket u\rrbracket \subseteq \overline{R} \implies k\llbracket z_r(u)\rrbracket = k\llbracket u\rrbracket = \overline{R}\\
\Longrightarrow\quad & \operatorname{ord}(z_r) = 1
\end{aligned}
$$

Sei nun umgekehrt $\operatorname{ord}(z_r) = 1$. Es ist $z_r(u) \in Q(k\llbracket x(u), y(u)\rrbracket)$

$$
\Longrightarrow\quad k(\!(u)\!) = k(\!(z_r(u))\!) \subseteq Q(k\llbracket x(u), y(u)\rrbracket) = Q(R) \implies u \in Q(R)
$$

Also ist $\{x, y\}$ eine primitive Parametrisierung. ▨

Wenn wir also voraussetzen, daß die vorgegebene Parametrisierung bei der Eingabe des Algorithmus `Param-HNE` *primitiv* ist, haben wir ein Kriterium, wann wir uns in der letzten Zeile der HNE befinden.

Beispiel: Von Parametrisierungen in Puiseux-Form oder solchen, deren HNE schon bekannt ist, läßt sich eindeutig sagen, ob sie primitiv sind oder nicht:

- Sei
$$x = T^n$$
$$y = a_{r_1}T^{r_1} + a_{r_2}T^{r_2} + \dots$$
eine Parametrisierung mit $1 \le r_1 < r_2 < \dots$ und $a_{r_i} \neq 0 \quad \forall\, i \ge 1$.
(x, y) ist genau dann primitiv, wenn $\operatorname{ggT}(\{n, r_1, r_2, \dots\}) = 1$ ist (siehe [Ca, 2.1.13]).

- Falls die Hamburger-Noether-Entwicklung schon bekannt ist, liefert uns Lemma 2.3 ein Kriterium, wann (x, y) primitiv ist.

Wir stellen nun eine Variante des Algorithmus `Param-HNE` vor, die terminiert:

Algorithmus `PrimParam-HNE`

Eingabe: $x, y \in k[\![T]\!]$, so daß (x, y) primitive Parametrisierung (irgendeiner Kurve) ist, und $\operatorname{ord}(x) \le \operatorname{ord}(y)$ (wir vereinbaren $\operatorname{ord}(0) := \infty$).

Ausgabe: Wenn
$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j} z_{j+1}\,, \quad j := 0, \dots, r$$
die (vollständige) HNE von (x, y) ist (mit $z_{-1} = y$), dann werden $h_0, \dots, h_{r-1}$, die a_{ji} für $1 \le i \le h_j$, $0 \le j \le r-1$ und die a_{ri} für $1 \le i \le \inf(\{i \,|\, a_{ri} \neq 0\})$ berechnet.

Schritt 0: $z_{-1} \leftarrow y, \quad z_0 \leftarrow x, \quad s \leftarrow 0$

Schritt 1: $i \leftarrow 1, \quad y_0 \leftarrow z_{s-1}$

Schritt 2:
Berechne $a_{si} \in k$ und $y_i \in k[\![T]\!]$ mit $\frac{y_{i-1}}{z_s} = a_{si} + y_i$ und $\operatorname{ord}_T(y_i) > 0$.

`falls` $a_{si} \neq 0$ `und` $\operatorname{ord}(z_s) = 1$ `dann STOP` (∗)
`falls` $y_{i-1} = 0$ `dann` setze $a_{sj} \leftarrow 0 \ \forall j > i$ und `STOP` (∗)

`falls` $\operatorname{ord}(y_i) \ge \operatorname{ord}(z_s)$
`dann` setze $i \leftarrow i + 1$ und wiederhole Schritt 2
`sonst` setze $h_s \leftarrow i$ (und weiter mit Schritt 3)

Schritt 3: $s \leftarrow s + 1, \quad z_s \leftarrow y_{h_{s-1}}$, gehe zurück nach Schritt 1.

Verifikation der Korrektheit: Der Algorithmus berechnet den Anfang der HNE richtig, denn er unterscheidet sich von `Param-HNE` nur um ein zusätzliches Abbruchkriterium (∗). Die letzte Zeile der HNE wird nach endlich vielen Schritten erreicht, d. h. `PrimParam-HNE` terminiert nach Lemma 2.3. Wird der Algorithmus wegen $\operatorname{ord}(z_s) = 1$ gestoppt, ist klar, daß genau die unter *Ausgabe* versprochenen Daten berechnet sind. Endet er dagegen, weil $y_{i-1} = 0$ ist, so sind alle weiteren $a_{si} = 0$, d. h. die HNE ist exakt berechnet. Insbesondere befinden wir uns also in der letzten Zeile, das bedeutet aber $\operatorname{ord}(z_s) = 1 \Rightarrow a_{si} = 0 \ \forall\, i \ge 1$.

Bemerkung: Aus $a_{ri} = 0 \ \forall\, i \ge 1$ folgt übrigens, daß schon $z_{r-1} = 0$ war. Das bedeutet aber $z_i = 0 \ \forall -1 \le i \le r-1$, also war $y = 0$ gegeben. Da nicht auch

$z_0 = x = 0$ sein kann ($(0,0)$ ist keine Parametrisierung), folgt $r = 0$. In diesem Fall lautet die Hamburger-Noether-Entwicklung also einfach

$$z_{-1} = 0\,z_0 + 0\,z_0^2 + \ldots = 0$$

Rechenbeispiele: Wir geben an dieser Stelle je ein Beispiel für eine abbrechende HNE und für eine ohne erkennbares Ende an.

1) Gegeben: $k = \mathbb{F}_7 = \mathbb{Z}/7\mathbb{Z}, \quad x, y \in \mathbb{F}_7[[t]]$:

$$\begin{aligned} x &= 2t^4 + 2t^5 + t^7 \\ y &= -2t^{10} - 3t^{11} - t^{13} - t^{14} + t^{15} + 3t^{16} + 2t^{17} \end{aligned}$$

Schritt 0: $z_{-1} := y,\ z_0 := x,\ s := 0\,;$ *Schritt 1:* $i := 1,\ y_0 = z_{-1}$

Schritt 2: $$\frac{y_0}{z_0} = \frac{y}{x} = \frac{t^{10}(-2 - 3t - t^3 - t^4 + t^5 + 3t^6 + 2t^7)}{t^4(2 + 2t + t^3)} = t^6(-1 + 3t - 3t^2 + 3t^3 + 2t^4) =: y_1, \quad \mathbf{a_{01} := 0}$$

$\mathrm{ord}(y_1) > \mathrm{ord}(z_0) > 1 \ \rightsquigarrow\ i := 2$ und wiederhole

Schritt 2: $$\frac{y_1}{z_0} = \frac{t^2(-1 + 3t - 3t^2 + 3t^3 + 2t^4)}{2 + 2t + t^3} = t^2(3 + 2t) =: y_2, \quad \mathbf{a_{02} := 0}$$

$\mathrm{ord}(y_2) < \mathrm{ord}(z_0) \ \rightsquigarrow\ \mathbf{h_0 := 2}$ und weiter mit

Schritt 3: $s := 1,\ z_1 := y_2 = t^2(3 + 2t)\,;$ *Schritt 1:* $i := 1,\ y_0 := z_0$

Die HNE sieht bis jetzt folgendermaßen aus:

$$\boxed{\begin{aligned} z_{-1} &= 0\,z_0 + 0\,z_0^2 + z_0^2 z_1 = z_0^2 z_1 \\ z_0 &= \cdots\cdots\cdots\cdots\cdots\cdots \end{aligned}}$$

Schritt 2: $$\frac{y_0}{z_1} = \frac{t^4(2 + 2t + t^3)}{t^2(3 + 2t)} = t^2(3 + t - 3t^2) =: y_1, \quad \mathbf{a_{11} := 0}$$

$\mathrm{ord}(y_1) = \mathrm{ord}(z_1) > 1 \ \rightsquigarrow\ i := 2$ und wiederhole

Schritt 2: $$\frac{y_1}{z_1} = \frac{3 + t - 3t^2}{3 + 2t} = 1 + 2t \ \rightsquigarrow\ \mathbf{a_{12} := 1}, \quad y_2 := 2t$$

$\mathrm{ord}(y_2) < \mathrm{ord}(z_1) \ \rightsquigarrow\ \mathbf{h_1 := 2}$ und weiter mit

Schritt 3: $s := 2,\ z_2 := y_2 = 2t\,;$ *Schritt 1:* $i := 1,\ y_0 := z_1$

Wir kennen nun eine weitere Zeile der HNE:

$$\boxed{\begin{aligned} z_{-1} &= z_0^2 z_1 \\ z_0 &= z_1^2 + z_1^2 z_2 \\ z_1 &= \cdots\cdots\cdots \end{aligned}}$$

Schritt 2: $$\frac{y_0}{z_2} = \frac{1}{2}\,t\,(3 + 2t) = t\,(-2 + t) =: y_1, \quad \mathbf{a_{21} := 0}$$

$\mathrm{ord}(y_1) = \mathrm{ord}(z_2) = 1$, d. h. wir haben die letzte Zeile erreicht, aber es ist $a_{21} = 0$, also rechnen wir weiter: $i := 2$ und wiederhole

Schritt 2: $\frac{y_1}{z_2} = \frac{1}{2}(-2+t) = -1-3t \rightsquigarrow \mathbf{a_{22}} := \mathbf{-1}, \quad y_2 := -3t$

$a_{22} \neq 0$, $\text{ord}(z_2) = 1 \rightsquigarrow$ `STOP`

Ergebnis: Der Algorithmus `PrimParam-HNE` liefert folgende HNE:

$$\boxed{\begin{array}{rcr} z_{-1} & = & z_0^2 z_1 \\ z_0 & = & z_1^2 + z_1^2 z_2 \\ z_1 & = & -z_2^2 + \ldots \end{array}}$$

Wir sehen allerdings sofort, daß die nächste Division eine konstante Potenzreihe geben würde, d. h. $y_3 = 0$ und die HNE bricht ab. Daher rechnen wir mit `Param-HNE` die Hamburger-Noether-Entwicklung noch fertig:

Schritt 2 (weiter): $\text{ord}(y_2) = \text{ord}(z_2)$, daher setze $i := 3$ und wiederhole

Schritt 2: $\frac{y_2}{z_2} = -\frac{3}{2} = \mathbf{2} =: \mathbf{a_{23}}, \quad y_3 := 0$ und die (exakte) HNE lautet:

$$\boxed{\begin{array}{rcr} z_{-1} & = & z_0^2 z_1 \\ z_0 & = & z_1^2 + z_1^2 z_2 \\ z_1 & = & -z_2^2 + 2\, z_2^3 \end{array}}$$

Aus dieser Hamburger-Noether-Entwicklung können wir gemäß Lemma 2.1 wieder eine Parametrisierung der Kurve durch Rückwärtseinsetzen gewinnen:

$$\begin{array}{rrcl} & z_1(T) & = & -T^2 + 2T^3 \\ \Longrightarrow & z_0(T) & = & z_1(T)^2 + z_1(T)^2 T = (-T^2+2T^3)^2(1+T) = T^4 - 3T^5 - 3T^7 \\ \Longrightarrow & z_{-1}(T) & = & z_0(T)^2 z_1(T) = (T^4 - 3T^5 - 3T^7)^2(-T^2+2T^3) \\ & & = & -T^{10} + T^{11} + 3T^{13} - 2T^{14} + T^{15} - 2T^{16} - 3T^{17} \end{array}$$

Lemma 2.1 besagt nun, daß die Potenzreihen (Polynome sogar, da die HNE glücklicherweise abbricht) $(z_0(T), z_{-1}(T))$ eine (primitive) Parametrisierung von R sind.

Probe: Ersetze T wieder durch $z_2 = 2t$, dann erhalten wir in der Tat

$$z_0(2t) = (2t)^4 - 3(2t)^5 - 3(2t)^7 = x, \quad z_{-1}(2t) = y$$

d. h. die beiden Parametrisierungen (x,y) und $(z_0(T), z_{-1}(T))$ sind *äquivalent.*

Beispiel 2) Gegeben: $k = \mathbb{F}_3, \quad x, y \in \mathbb{F}_3[\![t]\!]: \qquad x = t^3 + t^4, \quad y = t^5$

$z_{-1} := t^5, \quad z_0 := t^3 + t^4, \quad y_0 := z_{-1} = t^5$

$\frac{y_0}{z_0} = \frac{t^5}{t^3(1+t)} = \frac{t^2}{1+t} =: y_1, \quad \mathbf{a_{01}} := \mathbf{0}$

$\text{ord}(y_1) = 2 < 3 = \text{ord}(z_0) \rightsquigarrow \mathbf{h_0} := \mathbf{1}, \quad z_1 := y_1 = \frac{t^2}{1+t}, \quad y_0 := z_0 = t^3 + t^4$

$\frac{y_0}{z_1} = \frac{(t^3+t^4)(1+t)}{t^2} = (t+t^2)(1+t) = t - t^2 + t^3 =: y_1, \quad \mathbf{a_{11}} := \mathbf{0}$

$\text{ord}(y_1) = 1 < 2 = \text{ord}(z_1) \rightsquigarrow \mathbf{h_1} := \mathbf{1},\ z_2 := y_1 = t - t^2 + t^3,\ y_0 := z_1 = \frac{t^2}{1+t}$

$$\begin{aligned}
\frac{y_0}{z_2} &= \frac{t^2}{t(1-t+t^2)(1+t)} = \frac{t}{1+t^3} := y_1, \quad \mathbf{a_{21} := 0} \\
\frac{y_1}{z_2} &= \frac{t}{(1+t^3)t(1-t+t^2)} = \frac{1}{1-t+t^2+t^3-t^4+t^5} \\
&= \sum_{n\in\mathbb{N}} (t-t^2-t^3+t^4-t^5)^n = 1\ +t-t^2-t^3+t^4-t^5 \\
&\quad +t^2+t^3-t^4+t^5+O(t^6)\ +t^3+O(t^6)\ +t^4-t^5+O(t^6)\ +t^5+O(t^6) \\
&= 1+t+t^3+t^4+O(t^6) \ \rightsquigarrow\ \mathbf{a_{22} := 1}, \quad y_2 := t+t^3+t^4+O(t^6)
\end{aligned}$$

Wegen $a_{22} \neq 0$ und $\operatorname{ord}(z_2) = 1$ ist jetzt die Abbruchbedingung von `PrimParam-HNE` erfüllt. Dieser Algorithmus liefert also folgende Hamburger-Noether-Entwicklung:

$$\boxed{\begin{array}{rcl} z_{-1} &=& z_0 z_1 \\ z_0 &=& z_1 z_2 \\ z_1 &=& z_2^2 + \ldots \end{array}}$$

Weil die HNE nicht vollständig berechnet ist, können wir auch nur die Anfangsterme einer Parametrisierung wieder zurückgewinnen, aber die Genauigkeit dieser Parametrisierung läßt sich durch (endliche) Fortsetzung der HNE-Berechnung beliebig erhöhen:

Aus der bisherigen HNE erhalten wir $z_1(T) = T^2 + O(T^3)$

$$\left.\begin{array}{rrcl} \Rightarrow & z_0(T) &=& T^3 + O(T^4) \\ \Rightarrow & z_{-1}(T) &=& T^5 + O(T^6) \end{array}\right\} \text{ ist eine erste approximative Parametrisierung der Kurve.}$$

Wie man sieht, liefert diese HNE nur den jeweils ersten Anfangsterm. Wir setzen also die Berechnung der HNE fort:

$$\begin{aligned}
\frac{y_2}{z_2} &= \frac{t+t^3+t^4+O(t^6)}{t(1-t+t^2)} = (1+t^2+t^3+O(t^5))(1+t-t^3-t^4+O(t^6)) \\
&= 1+t+t^2+t^3+O(t^5) \quad \rightsquigarrow\ \mathbf{a_{23} := 1},\ y_3 = t+t^2+t^3+O(t^5)
\end{aligned}$$

Folglich ist jetzt $z_1 = z_2^2 + z_2^3 + \ldots$
die letzte Zeile der HNE, also $z_1(T) = T^2 + T^3 + O(T^4)$

$$\left.\begin{array}{rrcl} \Rightarrow & z_0(T) &=& T^3 + T^4 + O(T^5) \\ \Rightarrow & z_{-1}(T) &=& T^5 - T^6 + O(T^7) \end{array}\right\} \text{ ist die nächstbessere Näherung an eine Parametrisierung.}$$

Damit ist nun klar, daß die Parametrisierung, die uns Lemma 2.1 liefert, wiederum eine andere ist als die ursprünglich vorgegebene. Wenn wir die letzte Zeile der HNE fortsetzen, erhalten wir für jedes neu berechnete $a_{2,i}$ je einen weiteren Koeffizienten für $z_{-1}(T)$ und $z_0(T)$. Das ist nicht nur in diesem Beispiel der Fall, sondern unabhängig vom Aussehen der übrigen Zeilen. Näheres über die zu erwartende Genauigkeit der Parametrisierung ist in Lemma 5.2 und anschließendem Korollar zu erfahren (Seite 168f).

$$\frac{y_3}{z_2} = \frac{t+t^2+t^3+O(t^5)}{t(1-t+t^2)} = (1+t+t^2+O(t^4))(1+t-t^3+O(t^4))$$

$$
\begin{aligned}
&= 1-t-t^2+O(t^4) \quad \rightsquigarrow \mathbf{a_{24}:=1},\ y_4=-t-t^2+O(t^4)\\
\frac{y_4}{z_2} &= (-1-t+O(t^3))(1+t+O(t^3)) = -1+t-t^2+O(t^3)\\
&\rightsquigarrow \mathbf{a_{25}:=-1},\ y_5=t-t^2+O(t^3)\\
\frac{y_5}{z_2} &= (1-t+O(t^2))(1+t+O(t^3)) = 1+O(t^2) \ \rightsquigarrow \mathbf{a_{26}:=1},\ y_6=O(t^2)\\
\frac{y_6}{z_2} &= O(t) \quad \rightsquigarrow \quad \mathbf{a_{27}:=0}
\end{aligned}
$$

Hier brechen wir die Berechnung nun endgültig ab und kommen zu folgender Hamburger-Noether-Entwicklung:

$$
\boxed{\begin{aligned}
z_{-1} &= z_0 z_1\\
z_0 &= z_1 z_2\\
z_1 &= z_2^2+z_2^3+z_2^4-z_2^5+z_2^6+O\left(z_2^8\right)
\end{aligned}}
$$

Aus dieser wiederum gewinnen wir $z_1(T)=T^2+T^3+T^4-T^5+T^6+O(T^8)$ und daraus die Parametrisierung

$$
\begin{aligned}
z_0(T) &= T^3+T^4+T^5-T^6+T^7+O(T^9)\\
z_{-1}(T) &= T^5-T^6+T^9+O(T^{11})
\end{aligned}
$$

Es wäre natürlich hier nicht nötig, soviel Aufwand in eine möglichst genaue Parametrisierung zu investieren, da wir ja schon eine andere (und bessere, weil aus Polynomen bestehende) Parametrisierung kennen. Der eigentlich interessante Fall aber ist der, wenn die Kurve nicht durch eine Parametrisierung, sondern durch ihren lokalen Ring gegeben ist. Auch dann können wir eine HNE berechnen, durch deren Kenntnis wir eine Parametrisierung durch Rückwärtseinsetzen konstruieren können. In diesem Fall ist es erst wirklich sinnvoll, die Berechnung einer HNE weiter zu betreiben als bis zu dem Punkt, wo `PrimParam-HNE` abbricht.

2.3 HNE und das Aufblasen von Singularitäten

Bis jetzt haben wir Hamburger-Noether-Entwicklungen für ein System von Parametern $\{x,y\}$ (bzw. für eine Parametrisierung) betrachtet und festgestellt, daß diese (nach Wahl des transversalen Parameters) eindeutig bestimmt sind. Natürlich ist die Wahl eines solchen Systems $\{x,y\}$ eine willkürliche Angelegenheit. Wir untersuchen daher jetzt HNEs von Kurven ohne fixierte Basis und interessieren uns dabei – da uns hier die Eindeutigkeit verloren geht – für die Daten in einer HNE, die nur von der Kurve selbst abhängen, für die *Invarianten* der Hamburger-Noether-Entwicklung. Dies werden später auch diejenigen Daten sein, aus denen sich die im Abschnitt 1.3 definierten Invarianten (für irreduzible Kurvensingularitäten) herleiten lassen.

Ein wichtiger Schritt, um solche Invarianten zu bekommen, ist die Untersuchung, wie sich die HNE unter quadratischer Transformation verhält (vgl. Def. 1.16). Um quadratische Transformation machen zu können, müssen wir wieder annehmen, daß k algebraisch abgeschlossen ist. Zunächst besorgen wir uns neue Parameter für die quadratische Transformierte einer Kurve:

Lemma 2.4 *Sei $\{x,y\}$ Erzeugendensystem des maximalen Ideals von R, sei R_1 die quadratische Transformierte von R und*

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j}\, z_{j+1}\ , \quad j := 0, \ldots, r \tag{HN}$$

die (wohlgemerkt eindeutige) Hamburger-Noether-Entwicklung von $\{x,y\}$ mit $x = z_0$ und $y = z_{-1}$. Definiere

$$y_1 := \frac{y}{x} - a_{01}$$

Dann ist $\{x, y_1\}$ ein Erzeugendensystem des maximalen Ideals $\mathfrak{m}_1$ von R_1.

Beweis: Lemma 1.3 besagt gerade, daß $\mathfrak{m}_1$ erzeugt wird von $\{x, \frac{y}{x} - a\}$ für ein geeignetes Element $a \in k$.

$$\frac{y}{x} - a = \frac{z_{-1}}{z_0} - a = a_{01} - a + \sum_{i=2}^{h_0} a_{0i}\, z_0^{i-1} + z_0^{h_0-1} z_1$$

Damit $\frac{y}{x} - a$ überhaupt in $\mathfrak{m}_1$ liegt, muß also $a = a_{01}$ sein. ▨

Satz 2.5 *Mit den Bezeichnungen von Lemma 2.4 gilt: Ist* (HN) *die HNE von $\{x,y\}$ mit $x = z_0$ und $y = z_{-1}$, dann lautet die HNE von $\{x, y_1\}$*

(a) falls $h_0 > 1$

$$\begin{aligned} z_{-1} &= a_{02}\, z_0 + a_{03}\, z_0^2 + \ldots + a_{0h_0}\, z_0^{h_0-1} + z_0^{h_0-1} z_1 \\ z_{j-1} &= \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j}\, z_{j+1}\ , \quad j := 1, \ldots, r \end{aligned}$$

mit $x = z_0$ und $y_1 = z_{-1}$.

(b) falls $h_0 = 1$

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j}\, z_{j+1}\ , \quad j := 1, \ldots, r$$

mit $x = z_0$ und $y_1 = z_1$ (man beachte, daß diese HNE aus (HN) *durch Streichen der ersten Zeile entsteht).*

Der Beweis folgt direkt aus der Definition von y_1, wenn man die Werte von $z_0, \ldots, z_r$ beibehält. ▨

Dieser Satz hat eine wichtige Konsequenz, die auch die Frage beantwortet, welche Größen in der HNE einer Kurve invariant sind (in Teil (iii)):

Satz 2.6 (HNE und Multiplizitätensequenz)

Sei
$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j} z_{j+1}\,, \quad j := 0, \dots, r \tag{HN}$$
die HNE von $\{x, y\}$ *mit* $x = z_0$ *und* $y = z_{-1}$, *definiere ferner* $n_j := \underline{\nu}(z_j)$ *für* $j := 0, \dots, r$ *(*$\underline{\nu}$ *die zu* R *assoziierte Bewertung). Dann gilt:*

(i) Die verschiedenen Multiplizitäten, die beim sukzessiven Aufblasen von R *auftreten (Kette (1.3), S. 15), sind genau die* n_j, $0 \le j \le r$.

(ii) Jedes n_j *taucht exakt* h_j *mal in der Multiplizitätensequenz von* R *auf,* $0 \le j \le r, \quad h_r := \infty$.

(iii) Sei
$$z'_{j-1} = \sum_{i=1}^{h'_j} a'_{ji}\, z'^{\,i}_j + z'^{\,h'_j}_j z'_{j+1}\,, \quad j := 0, \dots, r' \tag{HN'}$$
eine beliebige Hamburger-Noether-Entwicklung von R, *sei* $n'_j := \underline{\nu}(z'_j)$.

Dann ist $r = r', \quad h_j = h'_j \quad \forall\, 0 \le j < r$ *und* $n_j = n'_j \quad \forall\, 0 \le j \le r$.

Beweis: (i) & (ii) zeigen wir mit Induktion über die minimale Anzahl quadratischer Transformationen $\boldsymbol{M(R)}$, die nötig sind, um R zu glätten. Ist $M(R) = 0$, dann ist die Kurve bereits glatt, d. h. $\underline{\nu}(x) = e(R) = 1 \;\Rightarrow\; k[\![x]\!] = R \ni y$

$$\Longrightarrow \quad \exists\, a_{01}, a_{02}, \ldots \in k: \quad y = a_{01}\, x + a_{02}\, x^2 + \ldots$$

Dies ist bereits die gesuchte Hamburger-Noether-Entwicklung. Es ist folglich $r = 0$, $n_0 = \underline{\nu}(z_0) = \underline{\nu}(x) = 1$ und per Konvention $h_0 = h_r := \infty$, was zu zeigen war.

Wir nehmen nun an, $M(R) > 0$ und (i) & (ii) gelten für alle Ringe R' mit der Eigenschaft $M(R') < M(R)$. Dazu gehört insbesondere die quadratische Transformierte R_1 von R; es folgt mit Satz 2.5:

(a) Falls $h_0 > 1$: $z_0, z_1, \ldots, z_r$ in der HNE von $\{x, y_1\}$ sind die gleichen wie in der HNE von $\{x, y\}$, also sind auch die n_j die gleichen geblieben. Nach Voraussetzung treten beim Aufblasen von R_1 die Multiplizitäten n_j auf, und zwar n_0 genau $(h_0 - 1)$ mal, die anderen n_j, $1 \le j \le r$, genau h_j mal. x ist in beiden Kurven transversal; das bedeutet, daß R und R_1 die gleiche Multiplizität haben: $e(R) = e(R_1) = n_0$, woraus die Behauptung folgt.

(b) Falls $h_0 = 1$: Dann hat R_1 die Multiplizität n_1, d. h. n_0 kommt in der Multiplizitätensequenz von R_1 nicht vor. Die anderen n_j treten h_j mal auf. Offenbar hat R die Multiplizität n_0, folglich tritt beim Desingularisieren von R die Multiplizität n_0 genau $(h_0 = 1)$ mal auf.

Die Aussage (iii) folgt direkt aus (i) & (ii), denn die Multiplizitätensequenz hängt nur von R ab. ▧

Korollar *Seien R und R' zwei irreduzible algebroide Kurven, sei*

$$\begin{array}{rclll} z_{j-1} & = & \displaystyle\sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j} z_{j+1}\,, & j := 0, \ldots, r & \textit{eine HNE von } R \textit{ und} \\ z'_{j-1} & = & \displaystyle\sum_{i=1}^{h'_j} a'_{ji}\, {z'_j}^i + {z'_j}^{h'_j} z'_{j+1}\,, & j := 0, \ldots, r' & \textit{eine HNE von } R'. \end{array}$$

Dann sind R und R' genau dann *äquiresolubel, wenn*

$$r = r', \quad h_j = h'_j \quad \forall\, 0 \le j < r \quad \textit{und} \quad n_j = n'_j \quad \forall\, 0 \le j \le r \quad \textit{sind.}$$

*Mit anderen Worten: Die h_j und n_j, $0 \le j \le r$, bilden ein **vollständiges System von Invarianten** für die Gleichauflösung.*

3 Berechnung der Hamburger-Noether-Entwicklung aus einer Gleichung

Die ersten beiden Abschnitte dieses Kapitels (3.1 & 3.2) folgen der Darstellung in der Dissertation von Marc Rybowicz [Ry]. Wir beschäftigen uns mit ebenen algebroiden Kurven, die durch eine Gleichung gegeben sind, d. h.:

Es sei k ein Körper und $f \in k[\![x,y]\!]$, so daß $R := k[\![x,y]\!] \,/\, \langle f \rangle$ eine ebene algebroide Kurve ist. Man beachte, daß dies impliziert:

- f ist keine Einheit und auch nicht Null
- f ist quadratfrei (weil R reduziert sein soll)

Definiere $\mathcal{R} := \overline{k}[\![x,y]\!] \,/\, \langle f \rangle$ – das ist eine ebene algebroide Kurve über einem algebraisch abgeschlossenen Körper. Sei $f = f_1 f_2 \cdots f_n$ die Zerlegung von f in irreduzible Faktoren über $\overline{k}$. Für jeden Zweig $\mathcal{R}_i := \overline{k}[\![x,y]\!] \,/\, \langle f_i \rangle$ von $\mathcal{R}$ gilt: Die Restklassen $[x]_{f_i}, [y]_{f_i}$ von x, y in $\mathcal{R}_i$ erzeugen das maximale Ideal von $\mathcal{R}_i$, und $\mathcal{R}_i$ besitzt eine Hamburger-Noether-Entwicklung in der Basis $\{[x]_{f_i}, [y]_{f_i}\}$.

Falls $\mathcal{R}$ (und damit $f \in \overline{k}[\![x,y]\!]$) irreduzibel ist (und k vollkommen), wird sich herausstellen, daß auch R selbst bereits eine HNE in der Basis $\{[x]_f, [y]_f\}$ besitzt mit gleichem ausgezeichneten transversalen Parameter (Satz 3.10). Diese HNE muß nach dem Korollar aus Satz 2.2 mit der HNE von $\mathcal{R}$ übereinstimmen, denn die HNE von R liefert eine gemeinsame Parametrisierung von R *und* $\mathcal{R}$ (unabhängig vom Körper ist $f(\mathrm{x}(z_r), \mathrm{y}(z_r)) = 0$). Ziel von 3.1 ist die Vorstellung eines Algorithmus zur Berechnung dieser HNE von R.

Falls $\mathcal{R}$ reduzibel ist, kann es Unterschiede zwischen R und $\mathcal{R}$ geben – z. B. kann die Anzahl der Zweige von R kleiner sein. Außerdem ist die Existenz einer HNE für alle Zweige nicht mehr gewährleistet. Wir beschränken uns im Teil 3.2 auf reduzible Kurven über algebraisch abgeschlossenen Körpern, d. h. wir verallgemeinern den Algorithmus von 3.1 zu einem Algorithmus, der die Hamburger-Noether-Entwicklungen von $\mathcal{R}$ berechnet. Genauer: Für jeden Zweig $\mathcal{R}_i$ von $\mathcal{R}$ werden wir die HNE von $\{[x]_{f_i}, [y]_{f_i}\}$ berechnen.

Im Teil 3.3 schließlich starten wir mit einem beliebigen Körper k und wechseln im Verlauf der Rechnung in eine (möglichst kleine) Körpererweiterung $\overline{k} \supseteq \widetilde{k} \supseteq k$, so daß f eine Zerlegung $f = f_1 f_2 \cdots f_n$ hat, $f_i \in \widetilde{k}[\![x,y]\!]$ und irreduzibel in $\overline{k}[\![x,y]\!]$ für alle i, und so daß die HNE von $\mathcal{R}_i = \overline{k}[\![x,y]\!] \,/\, \langle f_i \rangle$ gleich der HNE von $\widetilde{k}[\![x,y]\!] \,/\, \langle f_i \rangle$ ist.

Um $\widetilde{k}$ zu erhalten, adjungieren wir algebraische Elemente zu k. Da in Singular nur einfache algebraische Körpererweiterungen von $\mathbb{F}_p$ und $\mathbb{Q}$ verfügbar sind, wird die Hauptarbeit in Abschnitt 3.3 in der Berechnung eines primitiven Elements bestehen.

3.1 Irreduzibler Fall

In diesem Abschnitt sei k ein beliebiger Körper, $f \in k[\![x,y]\!]$ eine irreduzible Potenzreihe, keine Einheit und $R := k[\![x,y]\!] \,/\, \langle f \rangle$. Wir bezeichnen die Restklassen von x bzw. y in R mit x bzw. y, d. h. $R = k[\![\mathrm{x},\mathrm{y}]\!]$. $\mathfrak{m} = \langle \mathrm{x},\mathrm{y} \rangle$ ist das maximale Ideal von R. Die nachfolgend eingeführten Objekte verwendet unser Algorithmus:

3.1.1 Newtonpolygon und (quasi-)homogene Leitform

Definition 3.1 Ist $\quad f = \sum_{\alpha,\beta} c_{\alpha\beta}\, x^\alpha y^\beta \,,\quad$ dann heißt

$$\boldsymbol{D(f)} := \left\{ (\alpha,\beta) \in \mathbb{Z}_+^2 \;\middle|\; c_{\alpha\beta} \neq 0 \right\}$$

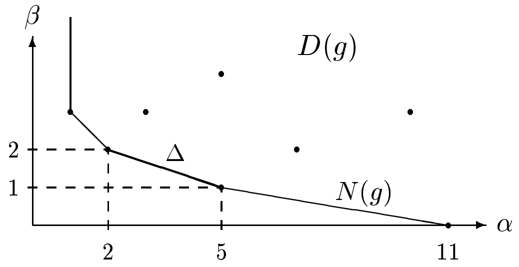
das **Newtondiagramm** von f. Das **Newtonpolygon** $\boldsymbol{N(f)}$ von f ist die Menge der Geradenstücke und Halbgeraden der konvexen Hülle von $D(f) + \mathbb{R}_+^2$, die nicht in den Koordinatenachsen enthalten sind. Die einzelnen Geradenstücke von $N(f)$ heißen **Seiten** des Newtonpolygons.

Die **Länge** bzw. **Höhe** einer Seite ist die Länge ihrer Projektion auf die α- bzw. β-Achse in $\mathbb{R}^2$.

Beispiel: $g = xy^3 + 2x^2y^2 - x^3y^3 + x^5y - x^5y^4 - 2x^7y^2 + x^{10}y^3 + x^{11}$

$$D(g) = \left\{ \begin{array}{l} (1,3),(2,2),(3,3),(5,1), \\ (5,4),(7,2),(10,3),(11,0) \end{array} \right\}$$

$N(g)$ besteht aus drei Geradenstücken und einer Halbgeraden. Sei Δ die Seite von $N(g)$ mit den Endpunkten $(2,2)$ und $(5,1)$, dann ist 3 die Länge und 1 die Höhe von Δ.



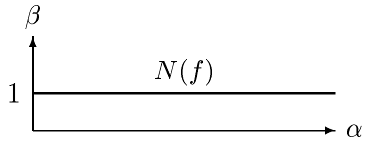
Bemerkung: Multipliziert man f mit einer Einheit aus $k[\![x,y]\!]$, so ändert sich das Newtonpolygon von f nicht, d. h. $N(g) = N(f)$ für alle Erzeuger g des Ideals $\langle f \rangle$. Daher heißt $N(f)$ auch das *Newtonpolygon der Kurve R*.

Wir betrachten das Newtonpolygon einer irreduziblen Potenzreihe f genauer:

Wenn das Newtonpolygon eine Halbgerade enthält, gilt $x \,|\, f$ oder $y \,|\, f$. Weil f irreduzibel ist, folgt daraus, daß es eine Einheit $u \in k[\![x,y]\!]^*$ gibt, so daß $f = xu$

oder $f = yu$ ist, also

$$\langle f \rangle = \langle x \rangle \quad \text{oder} \quad \langle f \rangle = \langle y \rangle$$



Wenn das Newtonpolygon keine Halbgerade enthält, gibt es Eckpunkte $A = (m, 0)$ und $B = (0, n)$ von $N(f)$ für geeignete $m, n \in \mathbb{N}$.

Lemma 3.1 *Ist $f \in k[\![x, y]\!]$ irreduzibel und sind $(m, 0), (0, n) \in D(f)$ Eckpunkte von $N(f)$, so ist*

$$\underline{\nu}(\mathrm{x}) = n \quad \textit{und} \quad \underline{\nu}(\mathrm{y}) = m$$

wobei x *und* y *die Restklassen von x bzw. y in $R = k[\![x, y]\!] \,/\, \langle f \rangle$ sind, $\underline{\nu}$ die zu R assoziierte Bewertung.*

Insbesondere läßt sich aus $N(f)$ direkt ablesen, welcher der Parameter $\mathrm{x}, \mathrm{y} \in R$ *transversal ist:*

$$\mathrm{x} \textit{ ist transversal} \iff n \leq m, \qquad \mathrm{y} \textit{ ist transversal} \iff m \leq n$$

Beweis: Sei $\mathcal{C} := \operatorname{Spec} R$ und $\mathcal{D} := \operatorname{Spec} k[\![x, y]\!] \,/\, \langle y \rangle$. Die Kurve $\mathcal{D}$ wird parametrisiert durch

$$x(T) = T, \quad y(T) = 0\,.$$

Daraus ergibt sich:

$$\underline{\nu}(\mathrm{y}) = \operatorname{mult}(\mathcal{D}, \mathcal{C}) = \operatorname{mult}(\mathcal{C}, \mathcal{D}) = \operatorname{ord}(f(T, 0)) = m$$

Die Aussage $\underline{\nu}(x) = n$ beweist man analog. Weil $\{\mathrm{x}, \mathrm{y}\}$ ein Erzeugendensystem von $\mathfrak{m}$ ist, ist der zweite Teil der Behauptung nichts weiter als die Anwendung der Definition von transversal. ▧

Satz 3.2 *Das Newtonpolygon einer irreduziblen Potenzreihe f besteht entweder aus einer Halbgeraden oder aus genau einer Seite.*

Beweis: Wir haben schon gesehen, daß $N(f)$ keine weiteren Seiten hat, wenn es eine Halbgerade enthält. Es bleibt der Fall zu betrachten, wenn $N(f)$ keine Halbgerade enthält.

O.B.d.A. sei x transversal (sonst vertausche x und y). Nach vorigem Lemma schneidet $N(f)$ die beiden Koordinatenachsen in $A = (m, 0)$ und $B = (0, n)$, wobei $n \leq m$ ist. Wir benutzen Induktion über $M(R)$, der Mindestzahl der zur Glättung von $R = k[\![x, y]\!] \,/\, \langle f \rangle$ benötigten quadratischen Transformationen.

I.A.: $M(R) = 0 \;\Rightarrow R$ ist schon regulär $\Rightarrow n = e(R) = 1$. Wenn aber $B = (0, 1)$ ist, kann $N(f)$ nur aus einer Seite bestehen.

I.V.: $M(R) > 0$ und das Lemma sei richtig für alle $f' \in k[\![x, y]\!]$ mit

$$M(k[\![x, y]\!] \,/\, \langle f' \rangle) < M(R).$$

Fall 1: $m > n$. Definiere $f_1 := x^{-n} f(x, xy)$, dann ist nach Lemma 1.7 die quadratische Transformierte $R_1 \cong k[[x,y]] \,/\, \langle f_1 \rangle$, d. h. das Newtonpolygon von R_1 ist gleich $N(f_1)$.

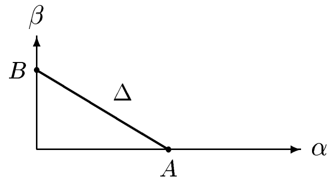
Definiere $T : \mathbb{Z}^2 \to \mathbb{Z}^2$ durch $(\alpha, \beta) \mapsto (\alpha + \beta - n,\, \beta)$. Dann ist

$$D(f_1) = T(D(f)) \qquad \text{und} \qquad N(f_1) = T(N(f))$$

(letzteres stimmt, weil für jeden Eckpunkt (α, β) von $N(f)$ gilt: $\beta <= n < m$).

$M(R_1) = M(R) - 1$, daher besteht $N(f_1)$ nach I.V. aus genau einer Seite. T ist eine affine Transformation von $\mathbb{Z}^2$, daher besteht auch das Urbild $N(f)$ aus genau einer Seite.

Fall 2: $m = n$. Weil der Untergrad von f gleich $e(R) = n$ ist, gibt es keine Punkte $(\alpha, \beta) \in D(f) : \alpha + \beta < n$. Also ist $N(f) = \{\overline{AB}\}$.



Zum Beweis der nächsten Aussage benötigen wir das *Henselsche Lemma* (Beweis z. B. in [ZaSa, VII §7]):

Lemma 3.3 (Hensel) *Sei A ein kompletter lokaler Ring mit maximalem Ideal $\mathfrak{m}$, $f \in A[y]$ normiert und $\deg f = r$. Wir bezeichnen mit $\overline{f} \in (A/\mathfrak{m})[y]$ das Polynom, das man erhält, wenn man die Koeffizienten von f durch ihre Restklassen in $A/\mathfrak{m}$ ersetzt. Dann gilt:*

$$\textit{Falls} \quad \alpha_1, \alpha_2 \in (A/\mathfrak{m})[y] \quad \textit{normiert sind,} \quad \deg \alpha_1 + \deg \alpha_2 = r,$$
$$1 \in \langle \alpha_1, \alpha_2 \rangle \qquad \textit{und} \qquad \alpha_1 \alpha_2 = \overline{f},$$

dann gibt es normierte Polynome

$$g_1, g_2 \in A[y], \quad \textit{s.d.} \qquad \deg g_i = \deg \alpha_i,$$
$$\overline{g_i} = \alpha_i \quad \textit{für } i = 1, 2 \quad \textit{und} \qquad g_1 g_2 = f.$$

Lemma 3.4 *Sei k ein algebraisch abgeschlossener Körper, $f \in k[[x,y]]$ irreduzibel und f_n die homogene Leitform von f, dann gibt es Elemente*

$$a, b \in k : \quad f_n = (ax + by)^n$$

Beweis: Nach eventuellem linearen Koordinatenwechsel sei f o.B.d.A. y-allgemein vom Grad n. Nach dem *Weierstraßschen Vorbereitungssatz* gibt es ein $u \in k[[x]]^*$ und ein $p \in (k[[x]])[y]$ normiert, $\deg p = n$, so daß $f = up$. Faßt man p als Potenzreihe in x, y auf, so hat p den gleichen Untergrad wie f, und es gibt ein $c \in k^*$, so daß $f_n = c\,p_n$ ist, p_n die homogene Leitform von p. Daher genügt es, die Aussage für ein Weierstraßpolynom zu zeigen.

Es sei also o.B.d.A.

$$\begin{array}{rrcl} & f & = & y^n + a_1(x)y^{n-1} + \ldots + a_n(x) \in (k[\![x]\!])\,[y] \\ \text{Definiere} & f'(x,y) & := & \dfrac{f(x,xy)}{x^n} = y^n + \dfrac{a_1(x)}{x}y^{n-1} + \ldots + \dfrac{a_n(x)}{x^n} \\ \Longrightarrow & f'(0,y) & = & f_n(1,y) \in k[y], \quad f' \text{ ist normiert und } \deg f'(0,y) = n. \end{array}$$

f' ist irreduzibel, $f'(0,y)$ ist das Bild von f' unter der Abbildung

$$\begin{array}{ccccc} k[\![x]\!][y] & \to & (k[\![x]\!]\,/\,\langle x\rangle)[y] & \cong & k[y] \\ \sum a_i y^i & \mapsto & \sum [a_i]_{\langle x\rangle} y^i & \mapsto & \sum a_i(0) y^i \end{array}$$

und zerfällt (wegen $k = \overline{k}$) vollständig in Linearfaktoren.

Nach dem *Henselschen Lemma* $\exists\, a \in k : f'(0,y) = (y+a)^n$, sonst ergäbe sich ein Widerspruch zur Irreduzibilität von f'. f_n ist homogen vom Grad n,

$$f_n(1,y) = (y+a)^n \quad \Rightarrow \quad f_n(x,y) = (y+ax)^n$$

Neue Voraussetzung: Ist unsere Kurve R durch f gegeben und $y \,|\, f$, dann ist die Restklasse $\mathrm{y} = [y]_f = 0$ und

$$z_{-1} = 0$$

ist die Hamburger-Noether-Entwicklung von R in der Basis $\{\mathrm{x},\mathrm{y}\}$, wobei $z_{-1} = \mathrm{y}$ und $z_0 = \mathrm{x}$ ist.

Gilt $x \,|\, f$, so hat R die gleiche HNE $\quad z_{-1} = 0, \quad$ diesmal mit $z_{-1} = \mathrm{x}$ und $z_0 = \mathrm{y}$.

Wir schließen diese beiden trivialen Fälle im folgenden aus. Außerdem nehmen wir ab jetzt immer an, daß x ein transversaler Parameter ist. Dann besteht $N(f)$ aus genau einer Seite, diese bezeichnen wir mit Δ. Die Endpunkte von Δ bezeichnen wir mit $A = (M,0)$ und $B = (0,N)$, m. a.W.: N sei die Höhe von Δ und M dessen Länge. Nach Voraussetzung ist $N \leq M$.

Definition 3.2 Ist $\;f = \sum\limits_{(\alpha,\beta)\in\mathbb{Z}_+^2} c_{\alpha\beta}\, x^\alpha y^\beta\;$ und Δ die Seite des Newtonpolygons, dann heißt

$$L := \boldsymbol{L(f)} := \sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, x^\alpha y^\beta \in k[x,y]$$

die **quasihomogene Leitform** von f. Diese hat für irreduzibles f wiederum eine ganz spezielle Form (analog zur Gestalt der homogenen Leitform in Lemma 3.4):

Lemma 3.5 *Mit f, M, N wie oben schreibe*

$$e := \mathrm{ggT}(M,N), \quad M' := M/e, \quad N' := N/e$$

Dann gibt es $c, \delta \in k^*$:

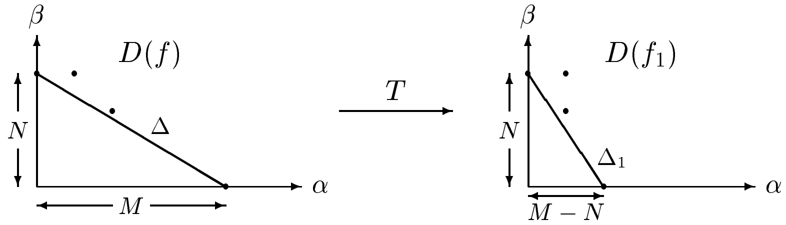
$$L(f) = c\left(y^{N'} - \delta x^{M'}\right)^e$$

Beweis: Wir benutzen wieder Induktion über $M(R)$:

I.A.: $M(R) = 0 \Rightarrow N = 1 \Rightarrow e = 1,\ N' = 1,\ M' = M$. Wähle $c := c_{0,1}$, $\delta := -c_{M,0}/c_{0,1}$, dann ist in der Tat $c \neq 0$, $\delta \neq 0$ und $L(f) = c\left(y - \delta x^{M'}\right)$, denn $\Delta \cap \mathbb{Z}_+^2 = \{A, B\}$.

I.V.: Wir betrachten eine Kurve R mit $M(R) > 0$ und nehmen an, daß die Behauptung für alle R' mit $M(R') < M(R)$ stimmt. Wir unterscheiden wieder zwei Fälle:

Fall 1: $M > N$. Die quadratische Transformierte R_1 von R wird durch $f_1 = x^{-N} f(x, xy)$ definiert. Das Newtonpolygon von f_1 ist wiederum das Bild von $N(f)$ unter der im Beweis zu Satz 3.2 definierten affinen Transformation T von $\mathbb{Z}^2$, daher besteht $N(f_1)$ aus einer Seite Δ_1 mit der Höhe N und der Länge $M - N$.



Es ist $\mathrm{ggT}(M - N, N) = \mathrm{ggT}(M, N) = e$, daher gilt nach I.V.:

$$\exists\, c, \delta \in k^* : L(f_1) = c\left(y^{N'} - \delta x^{M'-N'}\right)^e$$

Andererseits ist $L(f_1)(x, y) = x^{-N} L(f)(x, xy)$

$$\Longrightarrow\ L(f)(x,y) = x^N L(f_1)\left(x, \frac{y}{x}\right) = c\,x^N \left(\frac{y^{N'}}{x^{N'}} - \delta x^{M'-N'}\right)^e = c\left(y^{N'} - \delta x^{M'}\right)^e$$

Fall 2: $M = N$. In diesem Fall ist $L(f) = f_N$ die *homogene* Leitform von f. Aus Lemma 3.4 folgt $\exists\, a, b \in k : L(f) = (ax + by)^N$, und wegen $M = N$ gilt $a \neq 0 \neq b$. Daraus folgt die Behauptung. ▧

Nach diesen Vorbereitungen kommen wir zur Behandlung des Algorithmus.

Ausgangssituation: Wie auf Seite 47 vorausgesetzt sei $f \in k[[x, y]]$ irreduzibel, N die Höhe und M die Länge der einen Seite des Newtonpolygons von f, wobei $N \leq M$ ist.

Wir zerlegen M durch Division durch N mit Rest

$$M = QN + R\,, \qquad \text{wobei} \qquad 0 \leq R < N$$

und machen folgende Fallunterscheidung:

3.1.2 $R = 0$: **Transformation T_1**

Falls $R = 0$ ist, gibt es nach Lemma 3.5 $c, \delta \in k^*$:

$$L(f) = c\left(y - \delta x^Q\right)^N$$

Nach Lemma 3.1 gibt es eine Parametrisierung $(x(t), y(t))$ von f mit $\mathrm{ord}(x(t)) = N$ und $\mathrm{ord}(y(t)) = M$. Algorithmus `Param-HNE` legt nahe zu definieren

$$a_{0,i} := 0 \quad \text{für} \quad 1 \leq i < Q \qquad \text{und} \quad a_{0,Q} := \delta$$

Wir definieren außerdem die erste der beiden wichtigen Transformationen, auf denen der Algorithmus basiert:

Definition 3.3 Für $f \in k[\![x, y]\!]$, $\delta \in k$ und $Q \in \mathbb{N}$ definiere

$$\boxed{\boldsymbol{T_1}\ (f, \delta, Q)(x, y) := f\left(x, y + \delta x^Q\right)}$$

Unsere Definition der $a_{0,i}$ oben wird gerechtfertigt durch folgendes

Lemma 3.6 *Mit f, M, N, Q, δ wie oben setze $f_1 := T_1(f, \delta, Q)$. Dann gilt:*

(i) f_1 ist wieder irreduzibel, und das Newtonpolygon $N(f_1)$ ist entweder eine Halbgerade oder besteht aus einer Seite mit Höhe N und Länge $M_1 > M$. Eine Halbgerade kann nur auftreten, wenn vorher schon $N = 1$ war.

(ii) Ist $\left(z_0(t), z_{-1}(t)\right)$ eine Parametrisierung von f_1, so ist $\left(z_0(t),\ z_{-1}(t) + \delta z_0(t)^Q\right)$ eine Parametrisierung von f.

Oder anders ausgedrückt: Falls wir eine HNE von f_1 kennen:

$$\begin{array}{rcl} z_{-1} & = & a_{0,1}z_0 + \ldots + a_{0,h_0}z_0^{h_0} + z_0^{h_0}z_1 \\ z_0 & = & a_{1,2}z_1^2 + \ldots + a_{1,h_1}z_1^{h_1} + z_1^{h_1}z_2 \\ & \vdots & \end{array} \tag{3.1}$$

dann kennen wir auch eine HNE von f, nämlich

$$\begin{array}{rcl} z'_{-1} := z_{-1} + \delta z_0^Q & = & a_{0,1}z_0 + \ldots + (a_{0,Q} + \delta)z_0^Q + \ldots + a_{0,h_0}z_0^{h_0} + z_0^{h_0}z_1 \\ z'_0 := z_0 & = & a_{1,2}z_1^2 + \ldots + a_{1,h_1}z_1^{h_1} + z_1^{h_1}z_2 \\ & \vdots & \end{array} \tag{3.2}$$

Wir werden später[1] noch sehen, daß $a_{0,1} = \ldots = a_{0,Q} = 0$ in (3.1) ist, d. h. der Anfang von (3.2) lautet tatsächlich

$$z'_{-1} = \delta z_0^Q + z_0^{Q+1}(\ldots)$$

[1] Teil (iv) vom Beweis für den Algorithmus `HNE-irred-komplett`

Beweis: Zu (i): Wäre $f_1 = g_1 g_2$ für geeignete $g_i \in k[\![x,y]\!]$, so folgte

$$f(x,y) = f_1\left(x, y - \delta x^Q\right) = g_1\left(x, y - \delta x^Q\right) \cdot g_2\left(x, y - \delta x^Q\right) \quad \lightning$$

Als nächstes ist zu beachten, daß $T_1(L(f), \delta, Q) = cy^N$ ist, d. h. der Punkt $(M, 0)$ verschwindet aus dem Newtondiagramm von f_1 und $T_1(f - L(f), \delta, Q)$ trägt nichts bei zum Teil auf oder unterhalb der Seite von $N(f)$. Der Punkt $(0, N)$ bleibt dagegen in $D(f_1)$ erhalten. Also ist das neue Newtonpolygon $N(f_1)$ entweder eine Halbgerade oder es geht durch den Punkt $(0, N)$ und einen neuen Punkt $(M_1, 0)$ mit $M_1 > M$. Da f_1 irreduzibel ist, ist $N(f_1)$ eine Seite mit Höhe N und Länge M_1. Ist $N(f_1)$ eine Halbgerade, so gibt es eine Einheit $e \in k[\![x,y]\!]^*$, so daß $f_1 = ye$ ist ($f_1 = xe$ scheidet aus wegen $(0, N) \in D(f_1)$). Dann ist

$$f(x,y) = \left(y - \delta x^Q\right)\, e\left(x, y - \delta x^Q\right)\,, \quad \text{d. h.} \quad N = 1.$$

Zu (ii): Nach Voraussetzung ist

$$0 = f_1\big(z_0(t), z_{-1}(t)\big) = f\left(z_0(t), z_{-1}(t) + \delta z_0(t)^Q\right) \qquad \blacksquare$$

Bemerkung: Ist das Newtonpolygon von f_1 eine Halbgerade, so ist f_1 wie gesehen durch y teilbar und folglich $(t, 0)$ eine Parametrisierung von f_1. Es folgt nach (ii), daß $(t, \delta t^Q)$ eine Parametrisierung von f ist, d. h. die gesuchte Hamburger-Noether-Entwicklung lautet

$$z_{-1} = \delta z_0^Q$$

und wir sind fertig.

Ist $N(f_1)$ keine Halbgerade, so erfüllt f_1 wieder die Ausgangssituation, da ja $N \leq M < M_1$ ist; wir gehen also genauso vor, um die HNE von f_1 zu berechnen und setzen sie am Schluß nach der Vorschrift in (ii) zu einer HNE von f zusammen.

3.1.3 $R \neq 0$: **Transformation $\boldsymbol{T}$**

Falls $R \neq 0$ ist, besagt Lemma 3.1, daß der Algorithmus `Param-HNE` aus einer Parametrisierung (z_0, z_{-1}) von f eine HNE mit der ersten Zeile

$$z_{-1} = z_0^Q z_1$$

berechnet, wobei $\operatorname{ord}(z_1) = R$ ist.

Wir setzen daher

$$h_0 := Q, \quad a_{0,1} := a_{0,2} := \ldots := a_{0,Q} := 0$$

und definieren eine zweite Transformation:

Definition 3.4 Für $f \in k[\![x,y]\!]$ und $M, N \in \mathbb{N}$ definiere

$$\boldsymbol{U}(f, M, N)(x,y) := f\big(y, xy^Q\big)\,, \qquad \text{wobei} \quad M = QN + R, \quad 0 \leq R < N$$

dann ist $\bigl(y(t), x(t)\cdot y(t)^Q\bigr)$ eine Parametrisierung von f, falls $\bigl(x(t), y(t)\bigr)$ eine Parametrisierung von $U(f,M,N)$ ist. Mit dieser Transformation U erreichen wir jedoch nicht wieder die Ausgangssituation, denn $U(f,M,N)$ ist reduzibel: $y \mid f(y, xy^Q)$.

Aus diesem Grund verwenden wir für den Algorithmus nicht U, sondern T:

Definition 3.5 Für $f \in k[\![x,y]\!]$ irreduzibel und $M, N \in \mathbb{N}$ definiere

$$\boxed{\boldsymbol{T}(f,M,N)(x,y) := \frac{U(f,M,N)(x,y)}{y^{QN}} = \frac{f\left(y, xy^Q\right)}{y^{QN}}}\,, \quad Q \quad \text{wie oben.}$$

Man beachte die Ähnlichkeit zur klassischen quadratischen Transformation (Lemma 1.7): T ist nichts anderes als die Komposition von Q quadratischen Transformationen und einem Variablentausch $x \leftrightarrow y$.

Lemma 3.7 *Sei $f \in k[\![x,y]\!]$ irreduzibel und $N(f)$ bestehe aus einer Seite mit Höhe N und Länge M. Ferner sei $M = QN + R$ mit $0 < R < N$ und $f_1 := T(f,M,N)$. Dann gilt:*

(i) y^{QN} ist tatsächlich ein Teiler von $f\left(y, xy^Q\right)$, d. h. es ist $f_1 \in k[\![x,y]\!]$.

(ii) Man erhält das Newtondiagramm $D(f_1)$ aus $D(f)$ durch Anwendung folgender affiner Transformation

$$\mathcal{T} : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2, \quad \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \mapsto \begin{pmatrix} 0 & 1\\ 1 & Q\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} - \begin{pmatrix}0\\ QN\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha + Q(\beta - N)\end{pmatrix}$$

Das Newtonpolygon $N(f)$ wird ebenfalls von $\mathcal{T}$ auf das Newtonpolygon von f_1 abgebildet: $N(f_1) = \mathcal{T}(N(f))$

(iii) f_1 ist wieder irreduzibel, und $N(f_1)$ besteht aus einer Seite mit Höhe R und Länge N. Das heißt: f_1 erfüllt wegen $R < N$ wieder die Ausgangssituation.

(iv) Ist $(z_1(t), z_0(t))$ eine Parametrisierung von f_1, so ist $\bigl(z_0(t), z_0(t)^Q z_1(t)\bigr)$ eine Parametrisierung von f.

Oder anders ausgedrückt: Falls wir eine HNE von f_1 kennen:

$$\begin{aligned} [y]_{f_1} = z_0 &= a_{1,1}z_1 + \ldots + a_{1,h_1}z_1^{h_1} + z_1^{h_1}z_2 \\ [x]_{f_1} = z_1 &= \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\ \vdots \end{aligned} \tag{3.3}$$

dann ist $a_{1,1} = 0$ und wir erhalten eine HNE von f, indem wir einfach eine neue Zeile am Anfang der HNE von f_1 hinzufügen:

$$\begin{aligned} [y]_f = z_{-1} &= z_0^Q z_1 \\ [x]_f = z_0 &= a_{1,2}z_1^2 + \ldots + a_{1,h_1}z_1^{h_1} + z_1^{h_1}z_2 \\ z_1 &= \cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ &\ \vdots \end{aligned} \tag{3.4}$$

Beweis: Zu (i): Es sei $\displaystyle f(x,y) = \sum_{(\alpha,\beta)\in D(f)} c_{\alpha\beta}\, x^\alpha y^\beta,$ dann ist

$$f\left(y, xy^Q\right) = \sum_{(\alpha,\beta)\in D(f)} c_{\alpha\beta}\, x^\beta y^{\alpha+Q\beta} \tag{3.5}$$

Wir haben zu zeigen: $(\alpha,\beta) \in D(f) \quad \Rightarrow \quad QN \le \alpha + Q\beta.$

Per definitionem von M und N gilt: $(\alpha,\beta) \in D(f) \quad \Rightarrow \quad N\alpha + M\beta \ge MN$

$$\Longrightarrow \quad \alpha + Q\beta \ge M + \left(Q - \frac{M}{N}\right)\beta = M - \frac{R}{N}\beta$$

Ist $\beta \le N$, so folgt daraus $\quad \alpha + Q\beta \ge M - R = QN \quad \checkmark$
Ist $\beta \ge N$, so ist ohnehin $\quad \alpha + Q\beta \ge Q\beta \ge QN$ wegen $\alpha \ge 0$. $\checkmark$

Zu (ii) & (iii): Der erste Teil von (ii) ist eine triviale Folgerung aus (3.5).
Aus diesem folgt, daß $N(f)$ von $\mathcal{T}$ auf eine gerade Strecke mit den Endpunkten $(N, 0 + Q(N-N)) = (N,0)$ und $(0, M - QN) = (0,R)$ abgebildet wird. Angenommen es gebe einen Punkt $(\beta, \alpha + Q(\beta - N)) \in D(f_1)$ unterhalb dieser Strecke. Dann folgte

$$RN > R\beta + N(\alpha + Q(\beta - N)) = N\alpha + (R + QN)\beta - QN^2 = N\alpha + M\beta - QN^2$$

$$\Longleftrightarrow \; N\alpha + M\beta < (R + QN)N = MN$$

Das hieße aber, daß das Urbild (α,β) dieses Punktes unterhalb von $N(f)$ liegen müßte, also nicht zum Newtondiagramm $D(f)$ gehört. $\lightning$

Damit ist auch bewiesen, daß $N(f_1) = \mathcal{T}(N(f))$ aus einer Seite der Höhe R und Länge N besteht. Den Beweis der Irreduzibilität stellen wir noch zurück.

Zu (iv): $f_1(z_1(t), z_0(t)) = 0$

$$\Longrightarrow \quad 0 = z_0(t)^{QN} f_1(z_1(t), z_0(t)) = f\left(z_0(t), z_1(t) z_0(t)^Q\right)$$

Nun zur Behauptung $a_{1,1} = 0$. Wir können o.B.d.A. annehmen, daß $(z_1(t), z_0(t))$ eine primitive Parametrisierung von f_1 ist. Da $\left(z_0(t), z_0(t)^Q z_1(t)\right)$ dann eine primitive Parametrisierung von f ist, folgt daraus mit Lemma 3.1, daß

$$\operatorname{ord}(z_0(t)) = N \quad \text{und} \quad \operatorname{ord}\left(z_0(t)^Q z_1(t)\right) = M \quad \text{ist}$$

(f ist ja irreduzibel). Hieraus ergibt sich $\operatorname{ord}(z_1(t)) = M - QN = R$, und nach Voraussetzung ist $0 < R < N$. Also muß $a_{1,1} = 0$ sein, sonst würden wir aus der ersten Zeile von (3.3) erhalten: $R = \operatorname{ord}(z_1(t)) = \operatorname{ord}(z_0(t)) = N$.

Die Kenntnis dieser Parametrisierung von f verhilft uns zum Schluß auch zu einem Beweis der Irreduzibilität von f_1, nämlich auf folgende Weise:

Es gibt wenigstens einen irreduziblen Teiler g_1 von f_1 mit der gleichen Parametrisierung $(z_1(t), z_0(t))$, auf den sich Lemma 3.1 anwenden läßt. Weil wie eben gezeigt $\operatorname{ord}(z_1(t)) = R$ und $\operatorname{ord}(z_0(t)) = N$ ist, folgt wegen $R < N$, daß der Untergrad von g_1 gleich R ist. Wegen der Kenntnis des Newtonpolygons von f_1 (zweiter Teil von (iii)) wissen wir, daß auch f_1 den Untergrad R hat. Also hat f_1/g_1 den Untergrad Null, ist also eine Einheit. $\blacksquare$

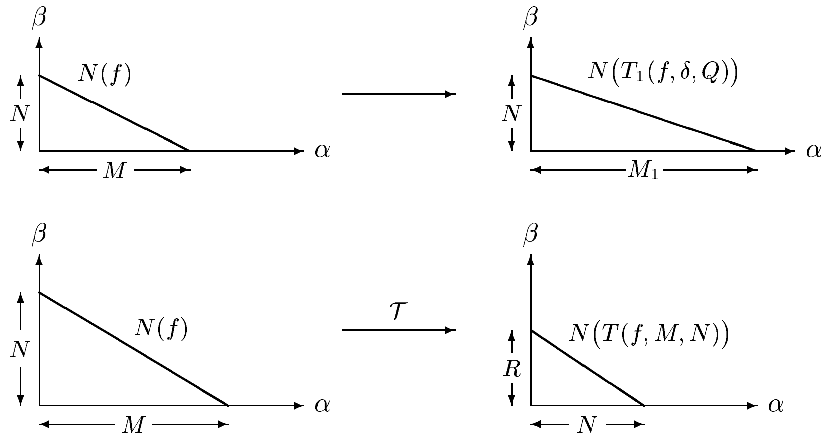


Abbildung 3.1: Transformationsverhalten des Newtonpolygons unter T_1 und T

3.1.4 Algorithmen

Mit Hilfe der beiden Transformationen T_1 und T können wir nun den Algorithmus zur Berechnung der HNE einer irreduziblen Potenzreihe (in einer ersten Fassung) formulieren:

Algorithmus `HNE-irred-komplett`

Eingabe: $f \in k[\![x, y]\!]$ irreduzibel, wobei k algebraisch abgeschlossener Körper (beliebiger Charakteristik) ist; es sei ferner $\mathrm{ord}_x(f(x, 0)) \geq \mathrm{ord}_y(f(0, y))$.

Ausgabe: Die Koeffizienten a_{ji} und die Exponenten h_i der Hamburger-Noether-Entwicklung von $k[\![x, y]\!] \,/\, \langle f \rangle$ mit $z_{-1} = [y]_f$ und $z_0 = [x]_f$.

Schritt 1: `falls` $y \mid f$ `dann` setze $a_{0,j} \leftarrow 0$ für alle $j \geq 1$ und `STOP`
`sonst` setze $s \leftarrow 0$, $q \leftarrow 0$ ($a_{s,q}$ ist das zuletzt berechnete Element)

Schritt 2: Berechne das Newtonpolygon $N(f)$: Es muß aus genau einer Seite Δ bestehen.
$M \leftarrow$ Länge(Δ), $N \leftarrow$ Höhe(Δ)

Schritt 3: Berechne Q, R mit $M = QN + R$ und $0 \leq R < N$.

`falls` $R = 0$ `dann` berechne $c, \delta \in k^* : L(f) = c\left(y - \delta x^Q\right)^N$
setze $a_{s,i} \leftarrow 0$ für $q < i < Q$ und $a_{s,Q} \leftarrow \delta$
setze $q \leftarrow Q$ und $f \leftarrow T_1(f, \delta, Q)$
`falls` $y \mid f$ `dann` setze $a_{s,j} \leftarrow 0$ für alle $j > q$ und `STOP`
`sonst` gehe zurück zu Schritt 2

`falls` $R \neq 0$ `dann`
setze $a_{s,i} \leftarrow 0$ für $q < i \leq Q$ und $h_s \leftarrow Q$
setze $q \leftarrow 0$, $s \leftarrow s + 1$ und $f \leftarrow T(f, M, N)$
gehe zurück zu Schritt 2

Bemerkung: Dieser Algorithmus berechnet wie `Param-HNE` die *vollständige* Hamburger-Noether-Entwicklung und hat darum eher theoretischen Wert – aus dem gleichen Grund wie bei `Param-HNE` kann `HNE-irred-komplett` im allgemeinen nicht terminieren.

Beweis, daß der Algorithmus das gewünschte Ergebnis hat:

(i) Die Voraussetzung $\operatorname{ord}_x(f(x,0)) \geq \operatorname{ord}_y(f(0,y))$ bedeutet gerade, daß $[x]_f$ transversal ist (Lemma 3.1). Daher gibt es genau eine HNE mit $z_{-1} = [y]_f$ und $z_0 = [x]_f$ (Satz 2.2).

(ii) In Schritt 2 gilt immer die Ausgangssituation: Zum einen ist f irreduzibel und das Newtonpolygon keine Halbgerade (dies wurde vorher getestet, bzw. im Fall $R \neq 0$ ist kein Test nötig – siehe 3.7 (iii)), zum anderen ist $M \geq N$. Man beachte, daß falls der Algorithmus schon einmal bei Schritt 3 war, sogar die strikte Ungleichung gilt (Lemmata 3.6 (i) & 3.7 (iii)): $M > N$.

(iii) Die Höhe des Newtonpolygons nimmt nicht zu, genauer: Im Fall $R = 0$ bleibt sie nach Lemma 3.6 (i) konstant, und im Fall $R \neq 0$ nimmt sie nach Lemma 3.7 (iii) echt ab.

Das bedeutet: Entweder der Algorithmus bricht ab, oder nach endlich vielen Schritten tritt nur noch der Fall $R = 0$ auf. `HNE-irred-komplett` liefert also ein Schema (HN) wie in der Definition auf Seite 25; es werden endlich viele $h_0, \ldots, h_{r-1}$ und $a_{s,i}$, $1 \leq i \leq h_s$ berechnet und (unendlich viele) $a_{r,i}$ für $i \in \mathbb{N}$. Außerdem ist $a_{s,1} = 0 \ \forall\, s > 0$ wegen der letzten Bemerkung von (ii).

(iv) *Im Fall $R = 0$ ist $Q > q$, im Fall $R \neq 0$ ist $Q \geq q$.*

Beweis: Das ist klar für $q = 0$. Wir brauchen also nur den Fall $q > 0$ betrachten, das heißt aber, daß vorher schon einmal Schritt 3 ausgeführt wurde und dabei $R = 0$ war. Es bezeichne M_1 die Länge und N_1 die Höhe des Newtonpolygons von f vor der letzten Transformation, sowie M und N die aktuell gültigen Werte. Dann gilt also $M_1 = qN_1$ (q ist ja das alte Q), $N_1 = N$ und $M_1 < M$ (die letzte Transformation war schließlich vom Typ T_1). Ferner gilt für die neuen Werte von Q und R: $M = QN + R$. Daraus folgt

$$qN = M_1 < M = QN + R$$

und daraus die Behauptung wegen $R < N$.

(v) Aus dem bisherigen folgt, daß der Algorithmus eine HNE (von irgendeiner Kurve) berechnet. Den Beweis, daß diese HNE tatsächlich die Hamburger-Noether-Entwicklung von f ist, erbringen wir im folgenden Satz:

Satz 3.8 *Die HNE, die der Algorithmus* `HNE-irred-komplett` *berechnet, ist die Hamburger-Noether-Entwicklung von f.*

Beweis: 1. Fall: Der Algorithmus bricht ab, d. h. liefert eine endliche HNE. Dann folgt die Behauptung sofort mit vollständiger Induktion aus 3.6 (ii) und 3.7 (iv): Irgendwann tritt ein f auf, das von y geteilt wird. Für dieses f ist

$$z_{-1} = 0 \qquad (\text{also} \quad a_{0,i} = 0 \quad \forall\, i \in \mathbb{N})$$

eine HNE, die der Algorithmus auch korrekt ermittelt (Induktionsanfang). Dieses f entstand durch Anwendung von endlich vielen Transformationen T und/oder T_1 aus der ursprünglich gegebenen Potenzreihe; der Induktionsschritt ist bereits in den beiden Lemmata 3.6 und 3.7 gemacht worden.

Daß der Algorithmus die HNE von $T_1(f, \delta, Q)$ bzw. $T(f, M, N)$ richtig zu einer HNE von f erweitert, wurde bereits im Schritt (iv) gezeigt ($Q \geq q$).

2. Fall: Der Algorithmus bricht nicht ab, d. h. uns fehlt der Induktionsanfang vom 1. Fall. Wir wissen aber immerhin, daß nur endlich oft eine Transformation vom Typ T auftritt. Es gibt also ein

$$i_0 \in \mathbb{N} : \forall\, i \geq i_0 \quad \text{ist } R = 0 \text{ im } i\text{-ten Durchlauf von Schritt 3.}$$

Es bezeichne f_{i_0} den Wert von f im i_0-ten Durchlauf von Schritt 2. Wir werden zeigen, daß wir durch Rückwärtseinsetzen in die HNE, die der Algorithmus für f_{i_0} berechnet, eine Parametrisierung von f_{i_0} erhalten. Dann können wir den Induktionsanfang im i_0-ten Durchlauf von Schritt 2 ansetzen.

Sei dazu f_i der Wert von f beim i-ten Durchlauf von Schritt 2, M_i die Länge und N_i die Höhe des Newtonpolygons $N(f_i)$. Für $i \geq i_0$ gilt

$$N_{i+1} = N_i, \quad M_{i+1} > M_i, \quad f_{i+1} = T_1(f_i, \delta_i, M_i/N_i) \quad \text{für } \delta_i \in k^* \text{ geeignet.}$$

Sei
$$z_{-1} = \sum_{j \in \mathbb{N}} a_{0j}\, z_0^j$$

die HNE, die der Algorithmus für f_{i_0} liefert (nach Voraussetzung ist diese HNE einzeilig). Definiere $Q_i := M_i/N_i$, dann ist $a_{0j} = \begin{cases} \delta_i & \text{falls } j = Q_i,\ i \geq i_0 \\ 0 & \text{sonst} \end{cases}$

Für $i \geq i_0$ definiere

$$x_i(t) := t, \quad y_i(t) := \sum_{j=1}^{Q_i} a_{0j}\, t^j$$

– das ist die „Parametrisierung“, die man durch Rückwärtseinsetzen in die HNE erhält, die im i-ten Iterationsschritt abgebrochen wurde. Dann gilt:

$$\forall\, i \geq i_0 :\ f_{i+1}(t, y) = f_i\left(t, y + \delta_i t^{Q_i}\right) \quad \text{für jedes beliebige} \quad y \in k[t]$$

Mit Induktion folgt daraus für beliebiges $i \geq i_0$:

$$f_{i+1}(t, 0) = f_{i_0}\left(t,\ \sum_{j=i_0}^{i} \delta_j\, t^{Q_j}\right) = f_{i_0}\left(t,\ \sum_{j=1}^{Q_i} a_{0j}\, t^j\right) = f_{i_0}(x_i(t), y_i(t))$$

$$\Longrightarrow \quad \forall\, i \geq i_0 :\ f_{i_0}(x_i(t), y_i(t)) \in \langle t \rangle^{M_{i+1}}$$

denn $(M_{i+1}, 0)$ ist der Schnittpunkt von $N(f_{i+1})$ mit der x-Achse des Newtondiagramms. Die Folge der M_i ist nach oben nicht beschränkt ($M_i < M_{i+1}$), folglich ist $\left(f_{i_0}(x_i(t), y_i(t))\right)_{i \in \mathbb{N}}$ eine Nullfolge (in der Krulltopologie).

$$\Longrightarrow \quad f_{i_0}\left(t,\ \sum_{j \in \mathbb{N}} a_{0j}\, t^j\right) = \lim_{i \to \infty} f_{i_0}(x_i(t), y_i(t)) = 0$$

Damit ist gezeigt, daß

$$\left(t,\ \sum_{j\in\mathbb{N}} a_{0j}\, t^j\right) \qquad \text{eine Parametrisierung von } f_{i_0} \text{ ist,}$$

das ist aber gerade die Parametrisierung, die man durch Rückwärtseinsetzen in die vom Algorithmus berechnete HNE von f_{i_0} gewinnt. Es bezeichne $\mathsf{HN}_{\mathsf{Alg}}$ die HNE, die `HNE-irred-komplett` aus f berechnet. Mit Induktion folgt aus 3.6 (ii) & 3.7 (iv) wie in Fall 1, daß die Parametrisierung, die man durch Rückwärtseinsetzen aus $\mathsf{HN}_{\mathsf{Alg}}$ gewinnt, eine Parametrisierung von f ist. Da die HNE dieser Parametrisierung wieder $\mathsf{HN}_{\mathsf{Alg}}$ sein muß, ist $\mathsf{HN}_{\mathsf{Alg}}$ die HNE von f (vgl. Korollar aus Satz 2.2). ▧

Der nächste Schritt besteht darin, wie bei `Param-HNE` ein Kriterium zu finden, wann die letzte Zeile der HNE erreicht ist, in der wir die Berechnung abbrechen können. Außerdem werden wir `HNE-irred-komplett` weiter präzisieren.

Lemma 3.9 *Genau dann treten im Algorithmus* `HNE-irred-komplett` *nur noch Transformationen vom Typ T_1 auf, wenn die Höhe N des Newtonpolygons gleich 1 ist.*

Beweis: Ist $N = 1$, so geht die Division in Schritt 3 natürlich auf, d. h. der Fall $R = 0$ tritt ein, d. h. eine Transformation T_1 wird gemacht. Dann bleibt die Höhe des Newtonpolygons konstant, also bleiben wir in allen folgenden Durchläufen im Fall $R = 0$.

Tritt nach endlich vielen Durchläufen ein f_{i_0} auf, auf das nur noch Transformationen vom Typ T_1 angewandt werden, dann besitzt f_{i_0} nach Satz 3.8 eine Parametrisierung $(t, \sum_i a_i t^i)$ für geeignete a_i. Diese Parametrisierung ist primitiv, daher folgt $N = 1$ aus Lemma 3.1. ▧

Nun zur konkreten Realisierung der Schritte 2 und 3 im Algorithmus:

Schritt 2: Da das Newtonpolygon nur aus einer Linie besteht, lassen sich M und N ganz einfach aus den Schnittpunkten von $N(f)$ mit den Koordinatenachsen bestimmen:

$$\text{Sei} \qquad f = \sum_{\alpha,\beta} c_{\alpha,\beta}\, x^\alpha y^\beta, \qquad \text{dann gilt}$$

$$M = \min\left\{\alpha \in \mathbb{Z} \mid c_{\alpha,0} \neq 0\right\}, \quad N = \min\left\{\beta \in \mathbb{Z} \mid c_{0,\beta} \neq 0\right\}$$

Schritt 3: *Zur Berechnung von c und δ:* $L(f)$ ist gegeben in der Form

$$L(f) = \sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, x^\alpha y^\beta$$

Die gesuchten c, δ erfüllen also folgende Bedingung:

$$\sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, x^\alpha y^\beta = c\left(y - \delta x^Q\right)^N = \sum_{i=0}^{N} c\binom{N}{i} y^{N-i}(-\delta)^i x^{Qi}$$

Daher ist $\boxed{c = c_{0,N}}$

Zur Berechnung von δ müssen wir eine Fallunterscheidung machen:

Fall 1: Für die Charakteristik des Körpers gelte: $\operatorname{char} k \nmid N$
(das ist insbesondere für $\operatorname{char} k = 0$ immer der Fall).

Dann folgt aus

$$c_{Q,N-1}\, x^Q y^{N-1} = -cN\delta\, x^Q y^{N-1}$$

$$\boxed{\delta = -\frac{c_{Q,N-1}}{N\, c_{0,N}}} \tag{3.6}$$

Fall 2: $\operatorname{char} k \mid N$: Dann sei $\operatorname{char} k =: p$ und $N =: n\, p^l$ mit $p \nmid n$.

Es gilt $c\left(y - \delta x^Q\right)^N = c\left(y^{p^l} - \delta^{p^l} x^{Q\,p^l}\right)^n$, woraus folgt:

$$\begin{aligned} c_{Q\,p^l,\,p^l(n-1)} x^{Q\,p^l} y^{p^l(n-1)} &= -cn\delta^{p^l} x^{Q\,p^l} y^{p^l(n-1)} \\ \Longrightarrow \quad \delta^{p^l} &= -\frac{1}{cn} c_{Q\,p^l,\,p^l(n-1)} \end{aligned}$$

Damit lautet die gesuchte Formel

$$\boxed{\delta = \sqrt[p^l]{-\frac{c_{Q\,p^l,\,N-p^l}}{n\, c_{0,N}}}} \tag{3.7}$$

In beiden Fällen ist δ durch die jeweils angegebene Formel eindeutig bestimmt – im Fall von (3.7) folgt das daraus, daß der sogenannte **Frobeniusendomorphismus**

$$\varphi : k \to k\,, \quad \varphi(a) := a^p, \quad p = \operatorname{char} k$$

injektiv ist.

Die Existenz von δ garantiert uns das Lemma 3.5, welches für algebraisch abgeschlossene Körper gilt. Die eben ermittelten Formeln legen es aber nahe, eine größere Menge von Körpern zuzulassen:

Definition 3.6 Ein Körper k heißt **vollkommen**, wenn $\operatorname{char} k = 0$ gilt oder der Frobeniusendomorphismus surjektiv ist.

Wir stellen fest:

Satz 3.10 *Ist k ein vollkommener Körper und ist $f \in k[[x,y]]$ irreduzibel in $\overline{k}[[x,y]]$, so existiert die Hamburger-Noether-Entwicklung von f bereits über k; genauer: Die HNE, die der Algorithmus* `HNE-irred-komplett` *berechnet, ist auch eine HNE von $k[[x,y]] \,/\, \langle f \rangle$ in der Basis $\{[x]_f, [y]_f\}$.*

Beweis: Der Algorithmus liefert eine HNE von $\mathcal{R} := \overline{k}[[x,y]] / \langle f \rangle$, da diese Kurve nach Voraussetzung irreduzibel ist. Wir haben nur noch zu zeigen, daß deren Koeffizienten $a_{s,i}$ Elemente von k sind, denn dann ist die aus der HNE gewonnene Parametrisierung $(x(t), y(t)) \in k[[t]] \times k[[t]]$ eine gemeinsame Parametrisierung von $\mathcal{R}$ und $R = k[[x,y]] / \langle f \rangle$. Jede Kurve, die sich parametrisieren läßt, besitzt eine HNE – `Param-HNE` liefert sie – und zwei Kurven mit der gleichen Parametrisierung besitzen die gleiche HNE (Korollar aus Satz 2.2).

Die in `HNE-irred-komplett` auftretenden Koeffizienten sind aber entweder Null oder eines der δ. Erfüllt δ die Gleichung (3.6), dann folgt $\delta \in k$ daraus, daß $c_{\alpha\beta}$ die Koeffizienten von f sind, andernfalls müssen wir zusätzlich voraussetzen, daß die p-te Wurzel nicht aus k herausführt. Das ist aber gerade äquivalent zur Surjektivität des Frobeniusendomorphismus. ▧

Für folgende Körper ist damit klar, wie die Berechnung der HNE konkret realisiert werden kann:

- Alle Körper der Charakteristik Null – wir brauchen nur Formel (3.6).
- Alle Körper $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ – wegen $a^p = a \quad \forall\, a \in \mathbb{F}_p$ ist

$$\sqrt[p^l]{a} = a \quad \forall\, a \in \mathbb{F}_p, \quad \forall\, l \in \mathbb{N}$$

- Alle endlichen Körper, denn in diesem Fall folgt die Surjektivität des Frobeniusendomorphismus schon aus der Injektivität. Alle endlichen Körper sind also vollkommen.

  Ist k ein endlicher Körper der Charakteristik p, aber $k \neq \mathbb{F}_p$, dann machen wir davon Gebrauch, daß $(k^*, \cdot)$ eine zyklische Gruppe ist und es ein $q \in \mathbb{N}$ gibt, so daß für die Anzahl der Elemente gilt: $|k| = p^q$ (vgl. [Ko, II.§1]). Sei a ein Erzeuger von $(k^*, \cdot)$ und $c \in k$ beliebig. Ist $c = 0$, so ist trivialerweise $\sqrt[p^l]{c} = 0$. Sonst gibt es ein $i \in \mathbb{N} : c = a^i$.

  Wir suchen nun ein $j \in \mathbb{N} : \left(a^j\right)^{p^l} = a^i$, das ist aber äquivalent zu

$$jp^l \equiv i \bmod (p^q - 1) \qquad \text{(kleiner Satz von Fermat).}$$

  p^l und $p^q - 1$ sind teilerfremd, daher gibt es $r, s \in \mathbb{N}$:

$$sp^l + r(p^q - 1) = 1 \quad \Longrightarrow \quad isp^l + ir(p^q - 1) = i$$

  Wähle also $j := is$, wobei s mit Hilfe des euklidischen Algorithmus berechnet werden kann.

Mit den bisherigen Erkenntnissen können wir eine neue Version von `HNE-irred-komplett` angeben, die den Anfang der HNE einer irreduziblen algebroiden Kurve berechnet und außerdem den Test, ob die gegebene Kurve irreduzibel ist, gleich mit ausführt.

Algorithmus `HNE-irreduzibel`

Eingabe: $f \in k[\![x,y]\!]$ quadratfrei, wobei k ein vollkommener Körper ist; es sei ferner $\operatorname{ord}_x(f(x,0)) \geq \operatorname{ord}_y(f(0,y))$.

Ausgabe:

- Falls $f \in \overline{k}[\![x,y]\!]$ irreduzibel ist:

  $$\text{Sei} \qquad z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j} z_{j+1}\,, \quad j := 0, \dots, r$$

  die (vollständige) HNE von f (mit $z_0 = [x]_f$), dann werden die $h_0, \dots, h_{r-1}$, die a_{ji} für $1 \leq i \leq h_j$, $0 \leq j \leq r-1$ und die a_{ri} für $1 \leq i \leq \inf(\{i \,|\, a_{ri} \neq 0\})$ (oder alle a_{ji}) berechnet (gleiche Ausgabe wie bei `PrimParam-HNE`).

- Falls $f \in \overline{k}[\![x,y]\!]$ reduzibel ist, wird eine Meldung „f reduzibel!“ ausgegeben.

Schritt 1: `falls` $y \,\big|\, f$ `dann` setze $a_{0,j} \leftarrow 0$ für alle $j \geq 1$ und `STOP`[2]
`sonst` setze $s \leftarrow 0$, $q \leftarrow 0$

Schritt 2: $M \leftarrow \operatorname{ord}_x(f(x,0))$, $N \leftarrow \operatorname{ord}_y(f(0,y))$
`falls` es einen Punkt in $D(f)$ unterhalb der Geraden durch die Punkte $(M,0)$ und $(0,N)$ gibt `dann STOP("f reduzibel!")`

Schritt 3: Berechne Q,R mit $M = QN + R$ und $0 \leq R < N$.

`falls` $R = 0$ `dann` berechne c, δ wie auf Seite 57
  `falls` $L(f) \neq c\left(y - \delta x^Q\right)^N$ `dann STOP("f reduzibel!")`
  setze $a_{s,i} \leftarrow 0$ für $q < i < Q$ und $a_{s,Q} \leftarrow \delta$
  setze $q \leftarrow Q$ und $f \leftarrow T_1(f, \delta, Q)$
  `falls` $y \,\big|\, f$ `dann` setze $a_{s,j} \leftarrow 0$ für alle $j > q$ und `STOP`[2]
  `falls` $N = 1$ `dann STOP`
  `sonst` gehe zurück zu Schritt 2
`falls` $R \neq 0$ `dann`
  setze $a_{s,i} \leftarrow 0$ für $q < i \leq Q$ und $h_s \leftarrow Q$
  setze $q \leftarrow 0$, $s \leftarrow s+1$ und $f \leftarrow T(f, M, N)$
  setze $M \leftarrow N$, $N \leftarrow R$
  wiederhole Schritt 3

Beweis der Korrektheit: Für den Fall $R \neq 0$ kann man auf Schritt 2 verzichten und stattdessen Lemma 3.7 anwenden. Falls f irreduzibel ist, ist der Algorithmus also äquivalent zu `HNE-irred-komplett` und liefert den Anfang der HNE. Der Algorithmus terminiert im irreduziblen Fall nach Lemma 3.9.

Umgekehrt: Wenn der Algorithmus in Schritt 3 an der Stelle $N = 1$ abgebrochen wird, muß das aktuelle f irreduzibel sein, da $\operatorname{ord}_y(f(0,y)) = 1$ ist. Wegen

[2] der Vollständigkeit halber kann man noch testen, ob f/y Einheit ist – wenn nicht, war f reduzibel; dennoch ist die berechnete HNE korrekt für einen Zweig von f

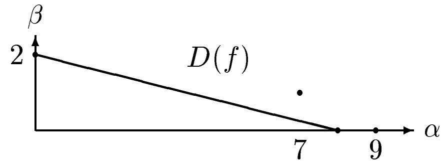
$T(gh) = T(g)T(h)$ und $T_1(gh) = T_1(g)T_1(h)$ folgt mit Induktion, daß dann auch das ursprüngliche f schon irreduzibel war. Wird der Algorithmus wegen $y \mid f$ beendet, kann man testen, ob f irreduzibel ist: Das ist genau dann der Fall, wenn der Punkt $(0,1)$ zu $D(f)$ gehört. Wenn man diesen Test ausführt, hat man folglich die Gewißheit: Falls f reduzibel über $\overline{k}$ ist, dann meldet `HNE-irreduzibel` dies auch, falls es terminiert.

Daß `HNE-irreduzibel` auch im reduziblen Fall terminiert, liegt daran, daß die gegebene Potenzreihe f als quadratfrei vorausgesetzt wird. Den Beweis dazu erbringen wir später mit dem Satz 3.22, welcher besagt, daß `HNE-reduzibel` für quadratfreies f terminiert. Dieser Satz beweist den gleichen Sachverhalt auch für `HNE-irreduzibel`, weil beide Algorithmen die gleichen Transformationen ausführen, solange `HNE-irreduzibel` nicht mit der Meldung „f reduzibel!“ abbricht.

Rechenbeispiele: 1) $k = \mathbb{F}_2, \quad f = y^2 + x^7y + x^8 + x^9$

Schritt 2: $M = 8, \quad N = 2$
Schritt 3: $8 = 4{\cdot}2{+}0 \rightsquigarrow Q = 4, \quad R = 0$

$L(f) = y^2 + x^8 = \left(y + x^4\right)^2 \rightsquigarrow \delta = 1$



$$\left(\text{oder mit Formel (3.7):} \quad N = 1 \cdot 2^1 \rightsquigarrow \delta = \sqrt[2^1]{-\frac{c_{8,0}}{1 \cdot c_{0,2}}} = \sqrt[2]{1} = 1\right)$$

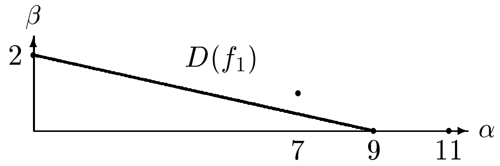
Wir erhalten daraus $a_{0,1} = a_{0,2} = a_{0,3} = 0$ und $a_{0,4} = 1$, d. h. der Anfang der HNE von f lautet

$$z_{-1} = z_0^4 + \ldots$$

$T_1(f,1,4) = f(x, y{+}x^4) = y^2{+}x^8{+}x^7y{+}x^{11}{+}x^8{+}x^9 = y^2{+}x^7y{+}x^9{+}x^{11} =: f_1$

Schritt 2: $M = 9, \quad N = 2$
Schritt 3: $9 = 4{\cdot}2{+}1 \rightsquigarrow Q = 4, \quad R = 1$

woraus wir erhalten: $\quad h_0 = 4$.



Nun ist die erste Zeile der HNE bekannt:

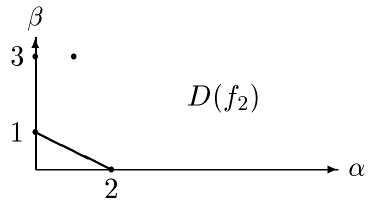
$$\begin{aligned} z_{-1} &= z_0^4 + z_0^4 z_1 \\ z_0 &= \ldots \end{aligned}$$

$T(f_1, 9, 2) = f_1(y, xy^4)/y^8 = \left(x^2y^8 + xy^{11} + y^9 + y^{11}\right)/y^8 = y{+}x^2{+}y^3{+}xy^3 =: f_2$

$M = 2, \quad N = 1$

Schritt 3: $2 = 2{\cdot}1{+}0 \rightsquigarrow Q = 2, \quad R = 0$

$L(f_2) = y + x^2 = \left(y + x^2\right)^N \rightsquigarrow \delta = 1$



$$\left(\text{oder mit Formel (3.6):} \quad \delta = -\frac{c_{2,0}}{1 \cdot c_{0,1}} = 1\right)$$

Wir erhalten daraus $a_{1,1} = 0$ und $a_{1,2} = 1$, d. h. der Anfang der HNE von f lautet

$$\begin{aligned} z_{-1} &= z_0^4 + z_0^4 z_1 \\ z_0 &= z_1^2 + \ldots \end{aligned}$$

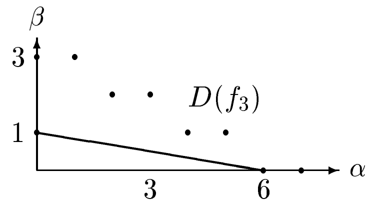
Das ist auch die Hamburger-Noether-Entwicklung, die `HNE-irreduzibel` zurückgibt, denn das Abbruchkriterium $N = 1$ ist eingetreten. Insbesondere hat der Algorithmus gezeigt: $f = y^2 + x^7 y + x^8 + x^9 \in \overline{\mathbb{F}_2}[[x, y]]$ ist irreduzibel.

In Anlehnung an `HNE-irred-komplett` kann man natürlich noch beliebig viele Iterationsschritte anhängen, um die Genauigkeit zu erhöhen, oder bis man feststellt, daß die HNE endlich ist. Wir berechnen noch den nächsten von Null verschiedenen Term von z_0:

$$\begin{aligned} T_1(f_2, 1, 2) &= y + x^2 + x^2 + (1 + x)\left(y + x^2\right)^3 \\ &= y + y^3 + xy^3 + x^2y^2 + x^3y^2 + x^4y + x^5y + x^6 + x^7 =: f_3 \end{aligned}$$

Schritt 2: $M = 6, \quad N = 1$
Schritt 3: $Q = 6, \quad R = 0$

$L(f_3) = y + x^6 = \left(y + x^6\right)^N \rightsquigarrow \delta = 1$



Daraus erhalten wir $a_{1,3} = a_{1,4} = a_{1,5} = 0$ und $a_{1,6} = 1$ und als neue HNE

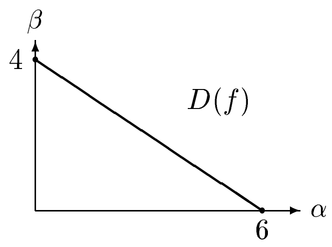
$$\begin{aligned} z_{-1} &= z_0^4 + z_0^4 z_1 \\ z_0 &= z_1^2 + z_1^6 + \ldots \end{aligned}$$

Die letzte Transformation T_1 zeigt ein typisches Verhalten für Polynome mit dünn besetztem Newtondiagramm: Die Anzahl der Monome „explodiert" geradezu. Das hat sich bei der Implementation in Hinblick auf die Rechenzeit als der alles bestimmende Faktor herausgestellt – f_4 hätte in diesem Beispiel bereits 18 Monome. In einem größeren Beispiel[3] hatte f den Grad 53 und bestand aus 102 Monomen, das transformierte Polynom am Ende von `HNE-irred` dagegen hatte den Grad 302 und bestand aus über 22000 Monomen.

Beispiel 2) $k = \mathbb{Q}, \quad f = y^4 - x^6$

Schritt 2: $M = 6, \quad N = 4$
Schritt 3: $Q = 1, \quad R = 2$

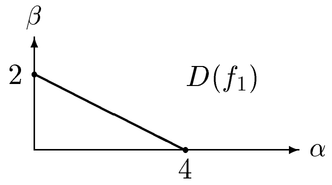
woraus wir erhalten: $a_{0,1} = 0, \quad h_0 = 1$



$$T(f, 6, 4) = f(y, xy)/y^4 = \left(x^4y^4 - y^6\right)/y^4 = x^4 - y^2 =: f_1$$

[3] $p_5 \in \mathbb{Q}[x, y]$ von Merle, vgl. S. 177

$M = 4, \quad N = 2 \rightsquigarrow Q = 2, \quad R = 0$
Schritt 3: $c = c_{0,2} = -1, \quad \delta = -\frac{c_{2,1}}{2c_{0,2}} = 0$



$c\left(y - \delta x^2\right)^2 \neq L(f_1) = x^4 - y^2$, daher stoppt der Algorithmus an dieser Stelle mit der Meldung „f ist reduzibel!“. Dies ist korrekt, denn $L(f_1) = -\left(y + x^2\right)\left(y - x^2\right)$, daher gibt es kein δ mit der Eigenschaft von Lemma 3.5. Dann erfüllt f_1 die Voraussetzung des Lemmas nicht, ist also reduzibel (in diesem Fall klar, da hier $f_1 = L(f_1)$ ist), dann ist aber auch f schon reduzibel gewesen. Und in der Tat gilt $f = \left(y^2 + x^3\right)\left(y^2 - x^3\right)$.

Betrachten wir den Fall $R \neq 0$ in Schritt 3 etwas genauer: Solange $R \neq 0$ ist, wird dieser Schritt wiederholt, und dabei auf M und N der euklidische Algorithmus angewandt. Das heißt, auf f wird eine Folge von Transformationen vom Typ T angewendet, ohne daß wir die Zwischenergebnisse wirklich brauchen. Aus diesem Grund können wir die einzelnen Transformationen auch zusammenfassen – das ist unser nächstes Ziel.

3.1.5 Zusammenfassung mehrerer Transformationen T

Bemerkung *(Zusammenhang zwischen HNE und euklidischem Algorithmus)*:
Sei $R := k[\![x,y]\!] \,/\, \langle f \rangle$ lokaler Ring einer irreduziblen algebroiden Kurve,
sei $\underline{\nu}([y]_f) = M, \quad \underline{\nu}([x]_f) = N$ und es gelte $N \leq M$.
Dann gibt es $h_0, \ldots, h_{l-1}, Q, N_1, \ldots, N_l \in \mathbb{N}, \quad z_{-1}, \ldots, z_l, y_1 \in \overline{R}$, so daß gilt:

$$\begin{array}{rcl} M &=& h_0 N + N_1 \\ N &=& h_1 N_1 + N_2 \\ N_1 &=& h_2 N_2 + N_3 \\ \multicolumn{3}{c}{\cdots\cdots\cdots\cdots\cdots} \\ N_{l-2} &=& h_{l-1} N_{l-1} + N_l \\ N_{l-1} &=& Q N_l \end{array} \qquad \begin{array}{rcl} [y]_f = z_{-1} &=& z_0^{h_0} z_1 \\ [x]_f = z_0 &=& z_1^{h_1} z_2 \\ z_1 &=& z_2^{h_2} z_3 \\ \multicolumn{3}{c}{\cdots\cdots\cdots} \\ z_{l-2} &=& z_{l-1}^{h_{l-1}} z_l \\ z_{l-1} &=& z_l^Q y_1 \end{array} \tag{3.8}$$

Dabei ist $N > N_1 > N_2 > \ldots > N_l, \quad N_l = \mathrm{ggT}(M,N)$ und y_1 Einheit in $\overline{R}$.

Es ist also möglich, z_{-1} und z_0 als Monome in z_{l-1} und z_l zu schreiben.

Wir suchen nun $a, b, c, d \in \mathbb{N}$: $\begin{array}{rcl} z_{-1} &=& z_{l-1}^a \, z_l^b \\ z_0 &=& z_{l-1}^c \, z_l^d \end{array}$

Lemma 3.11 *Mit den Bezeichnungen von oben und* $M' := M/\mathrm{ggT}(M,N)$, $N' := N/\mathrm{ggT}(M,N)$ *definieren wir* $\sigma, \tau \in \mathbb{Z}$ *durch:*

$$(3.9) \quad \left\{ \begin{array}{l} N'\sigma - M'\tau = (-1)^l \\ 0 < \sigma \leq M' \\ 0 \leq \tau \leq N' \end{array} \right.$$

Dann gilt:
$$\begin{array}{rcl} z_{-1} &=& z_{l-1}^{\sigma} \, z_l^{M' - \sigma Q} \\ z_0 &=& z_{l-1}^{\tau} \, z_l^{N' - \tau Q} \end{array}$$

Beweis: Mit Induktion sieht man, daß a, b, c, d definiert durch

$$\mathfrak{A} := \begin{pmatrix} h_0 & 1 \\ 1 & 0 \end{pmatrix} \cdots \begin{pmatrix} h_{l-1} & 1 \\ 1 & 0 \end{pmatrix} =: \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

die gesuchten Exponenten sind. Wir definieren desweiteren eine Hilfsmatrix

$$\mathfrak{B} := \begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -h_{l-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -h_0 \end{pmatrix}$$

für die wir folgendes zeigen werden:

$$\left.\begin{array}{l} \text{Es gibt} \quad \sigma', \tau' \in \mathbb{Z} \quad \text{mit} \\ \qquad (\diamondsuit) \quad \begin{cases} M'\tau' + N'\sigma' = (-1)^l \\ 0 < \sigma' \le M' \\ 0 \le -\tau' \le N' \end{cases} \\ \qquad \text{und} \qquad \mathfrak{B} = (-1)^l \begin{pmatrix} \tau' & \sigma' \\ N' & -M' \end{pmatrix} \end{array}\right\} \tag{3.10}$$

Indem wir $\sigma := \sigma'$ und $\tau := -\tau'$ wählen, erhalten wir dann nämlich

$$\begin{aligned} \mathfrak{B} \begin{pmatrix} \sigma & M' - \sigma Q \\ \tau & N' - \tau Q \end{pmatrix} &= (-1)^l \begin{pmatrix} \tau' & \sigma' \\ N' & -M' \end{pmatrix} \begin{pmatrix} \sigma' & M' - \sigma' Q \\ -\tau' & N' + \tau' Q \end{pmatrix} \\ &= (-1)^l \begin{pmatrix} 0 & \tau' M' + \sigma' N' \\ N'\sigma' + M'\tau' & -N'\sigma' Q - M'\tau' Q \end{pmatrix} \\ &= (-1)^{2l} \begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix} = \mathfrak{B} \cdot \mathfrak{A} \end{aligned}$$

$\det \mathfrak{B} = (-1)^{l+1}$, daher ist $\mathfrak{B}$ invertierbar.

$$\Longrightarrow \quad \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \mathfrak{A} = \begin{pmatrix} \sigma & M' - \sigma Q \\ \tau & N' - \tau Q \end{pmatrix}, \quad \text{q.e.d.}$$

Aus $(\diamondsuit)$ für σ', τ' folgt, daß σ und τ (3.9) erfüllen.

Angenommen $\sigma_1, \tau_1 \in \mathbb{Z}$ erfüllen (3.9) ebenfalls. Dann folgt

$$\begin{aligned} & N'(\sigma_1 - \sigma) = M'(\tau_1 - \tau), \quad |\sigma_1 - \sigma| \le M', \quad |\tau_1 - \tau| \le N' \\ \Longrightarrow \;& M' \mid N'(\sigma_1 - \sigma) \quad \Longrightarrow \quad M' \mid (\sigma_1 - \sigma), \quad \text{da} \quad \mathrm{ggT}(M', N') = 1 \\ \Longrightarrow \;& |\sigma_1 - \sigma| \in \{0, M'\} \quad \Longrightarrow \quad |\tau_1 - \tau| \in \{0, N'\} \end{aligned}$$

Wäre $|\sigma_1 - \sigma| = M'$, o.B.d.A. $\sigma_1 - \sigma = M'$, so folgte $\tau_1 - \tau = N'$
und weiter $\sigma_1 = M'$, $\sigma = 0$, $\tau_1 = N'$, $\tau = 0$.

Dann wäre jedoch $(-1)^l = N'\sigma - M'\tau = 0$ $\lightning$

Also ist $|\sigma_1 - \sigma| = 0 \Longrightarrow \tau_1 - \tau = 0$,

woraus folgt, daß σ und τ durch (3.9) eindeutig bestimmt sind (wir haben die Eindeutigkeit sogar für den Fall gezeigt, daß man $0 \le \sigma \le M'$ zuläßt).

Es bleibt noch, die Behauptung (3.10) zu zeigen. Dies erfolgt mit vollständiger Induktion über l:

I.A.: Für $l = 0$ ist die Behauptung trivial, denn dann ist $M = QN$ in (3.8) und $\mathfrak{B} = \begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix}$.

I.S.: Es seien M' und N' gegeben mit $\operatorname{ggT}(M', N') = 1$ und

$$\begin{aligned} M' &= h_0 N' + N_1' \\ N' &= h_1 N_1' + N_2' \\ N_1' &= h_2 N_2' + N_3' \\ \dots &\vdots \dots\dots \\ N_{l-2}' &= h_{l-1} N_{l-1}' + 1 \\ N_{l-1}' &= Q \end{aligned}$$

für ein $l > 0$.

Nach Induktionsvoraussetzung gibt es $\tilde{\sigma}, \tilde{\tau} \in \mathbb{Z}$ mit

$$N'\tilde{\tau} + N_1'\tilde{\sigma} = (-1)^{l-1}, \quad 0 < \tilde{\sigma} \le N', \quad 0 \le -\tilde{\tau} \le N_1' \qquad \text{und}$$

$$\begin{pmatrix} 0 & 1 \\ 1 & -Q \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -h_{l-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & -h_1 \end{pmatrix} = (-1)^{l-1} \begin{pmatrix} \tilde{\tau} & \tilde{\sigma} \\ N_1' & -N' \end{pmatrix}$$

$$\begin{aligned} \Longrightarrow \quad \mathfrak{B} &= (-1)^{l-1} \begin{pmatrix} \tilde{\tau} & \tilde{\sigma} \\ N_1' & -N' \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & -h_0 \end{pmatrix} \\ &= (-1)^{l-1} \begin{pmatrix} \tilde{\sigma} & \tilde{\tau} - \tilde{\sigma} h_0 \\ -N' & N_1' + N' h_0 \end{pmatrix} = (-1)^l \begin{pmatrix} -\tilde{\sigma} & \tilde{\sigma} h_0 - \tilde{\tau} \\ N' & -M' \end{pmatrix} \end{aligned}$$

Wir definieren also $\quad \tau' := -\tilde{\sigma}, \quad \sigma' := \tilde{\sigma} h_0 - \tilde{\tau}, \quad$ denn dann ist

$$\mathfrak{B} = (-1)^l \begin{pmatrix} \tau' & \sigma' \\ N' & -M' \end{pmatrix} \qquad \text{und}$$

$$M'\tau' + N'\sigma' = -M'\tilde{\sigma} + N'(\tilde{\sigma} h_0 - \tilde{\tau}) = -N_1'\tilde{\sigma} - N'\tilde{\tau} \underset{\text{IV}}{=} -(-1)^{l-1} = (-1)^l$$

$$0 < \underbrace{\tilde{\sigma}}_{-\tau'} \le N' \qquad \Longrightarrow \qquad \tau' < 0$$

$$\Longrightarrow \quad 0 < \tilde{\sigma} h_0 \le \underbrace{\tilde{\sigma} h_0 - \tilde{\tau}}_{\sigma'} \underset{\text{IV}}{\le} h_0 N' + N_1' = M'$$

Damit ist auch (3.10) bewiesen. ▧

Wir definieren also eine neue Transformation:

Definition 3.7 Seien $f \in k[\![x, y]\!]$, $M', N' \in \mathbb{N}$ und seien $\sigma, \tau \in \mathbb{Z}$ durch (3.9) bestimmt. Dann vereinbaren wir

$$\boxed{\boldsymbol{T}^*(f, M', N')(x, y) := f\left(x^{N' - \tau Q} y^{\tau},\ x^{M' - \sigma Q} y^{\sigma}\right)}$$

Bemerkung: Diese Transformation induziert einen Endomorphismus auf $\mathbb{Z}^2$,

$$\mathcal{T}^* : \binom{\alpha}{\beta} \mapsto \begin{pmatrix} N' - \tau Q & M' - \sigma Q \\ \tau & \sigma \end{pmatrix} \binom{\alpha}{\beta}$$

der nach Lemma 3.11 $\mathbb{Z}_2^+$ in sich abbildet. Wir untersuchen $\mathcal{T}^*(N(f))$:

Es bezeichnen wie gehabt $A = (M, 0)$ und $B = (0, N)$ die Endpunkte des Newtonpolygons $N(f)$, sowie $e := \mathrm{ggT}(M, N)$, $M' := M/e$, $N' := N/e$.

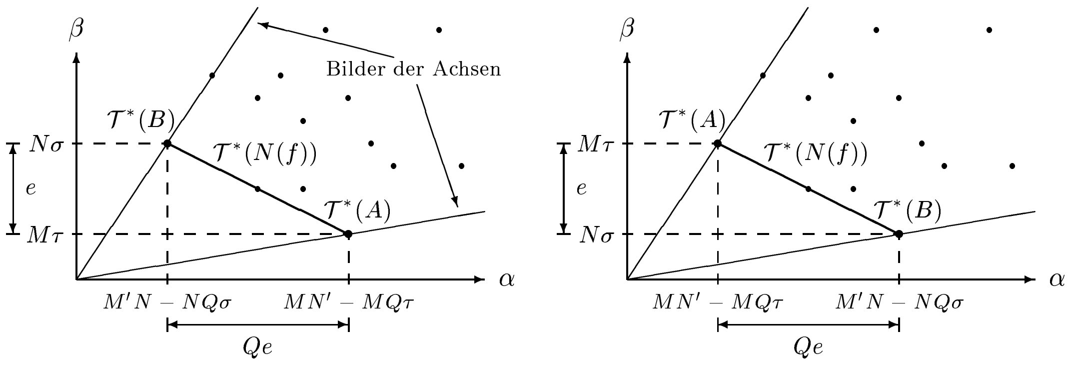
$$\begin{aligned} \mathcal{T}^*(A) &= \begin{pmatrix} N' - \tau Q & M' - \sigma Q \\ \tau & \sigma \end{pmatrix} \binom{M}{0} = \binom{MN' - MQ\tau}{M\tau} \\ \mathcal{T}^*(B) &= \begin{pmatrix} N' - \tau Q & M' - \sigma Q \\ \tau & \sigma \end{pmatrix} \binom{0}{N} = \binom{M'N - NQ\sigma}{N\sigma} \end{aligned}$$

$$\begin{array}{lll} \text{Falls} \quad N'\sigma - M'\tau = 1 & \Longrightarrow & N\sigma - M\tau = e > 0 \\ \text{sonst ist} & & N\sigma - M\tau = -e < 0 \end{array}$$

Außerdem gilt:

$$\begin{aligned} (MN' - MQ\tau) - (M'N - NQ\sigma) &= e(M'N' - M'Q\tau - M'N' + N'Q\sigma) \\ &= Qe(N'\sigma - M'\tau) = \pm Qe \end{aligned}$$

Je nachdem ob l gerade oder ungerade ist, erhalten wir also eines der beiden Bilder für $\mathcal{T}^*(D(f))$:



Um wieder die Ausgangssituation herzustellen, müssen wir noch das Newtondiagramm verschieben, d. h. $T^*(f)$ durch geeignete Potenzen von x und y dividieren. Die richtigen Werte lassen sich direkt an den beiden Bildern ablesen:

Lemma 3.12 *Sei $f \in k[[x, y]]$ irreduzibel, $N(f)$ bestehe aus einer Seite der Höhe N und Länge M, und es gelte (3.8) und (3.9). Wir wählen*

$$\begin{aligned} s &:= MN' - Q \cdot \max\{N\sigma, M\tau\} \\ t &:= \min\{N\sigma, M\tau\} \end{aligned}$$

Dann hat

$$f^* := \frac{T^*(f, M', N')}{x^s y^t}$$

folgende Eigenschaften:

(i) $x^s y^t \mid T^*(f, M', N')$, *d. h.* $f^* \in k[[x, y]]$

(ii) $N(f^*)$ *besteht aus einer Seite der Höhe* $e = \mathrm{ggT}(M, N)$ *und der Länge* Qe

(iii) $f^* = T(T(\dots T(f, M, N)\dots, N_{l-3}, N_{l-2}), N_{l-2}, N_{l-1})$

(iv) f^* *ist irreduzibel.*

Beweis: (i) & (ii) folgen direkt aus der vorangegangenen Bemerkung, denn s und t sind so gewählt, daß $\mathcal{T}^*(A)$ und $\mathcal{T}^*(B)$ auf die Koordinatenachsen verschoben werden. $\mathcal{T}^*$ ist eine lineare Abbildung, daher ist $\mathcal{T}^*(N(f))$ wieder eine Gerade, und es liegen keine Punkte von $D(T^*(f, M', N'))$ unterhalb von $\mathcal{T}^*(N(f))$. Länge und Höhe von $\mathcal{T}^*(N(f))$ haben wir schon berechnet; diese ändern sich bei einer anschließenden Translation nicht mehr.

Zu (iii): Wir zeigen mit Induktion über l, daß

$$T^*(\cdot, M', N') = U(\cdot, N_{l-2}, N_{l-1}) \circ \dots \circ U(\cdot, N, N_1) \circ U(\cdot, M, N) \tag{3.11}$$

ist, d. h. $\begin{pmatrix} N' - \tau Q & M' - \sigma Q \\ \tau & \sigma \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & h_{l-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & h_1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & h_0 \end{pmatrix}$

Falls $l = 0 \quad \Rightarrow \quad N' = 1, \quad Q = M', \quad \sigma = 1, \quad \tau = 0$

$$\Rightarrow \quad \begin{pmatrix} N' - \tau Q & M' - \sigma Q \\ \tau & \sigma \end{pmatrix} = \begin{pmatrix} 1 - 0 & M' - M' \\ 0 & 1 \end{pmatrix} = \mathbb{1}$$

Sei $l > 0$ und $\begin{pmatrix} N_1' - \tau Q & N' - \sigma Q \\ \tau & \sigma \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & h_{l-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & h_1 \end{pmatrix}$,
wobei gelte $N_1'\sigma - N'\tau = (-1)^{l-1}, \quad 0 < \sigma \le N', \quad 0 \le \tau \le N_1'$.

$$\begin{aligned} \Longrightarrow \quad \begin{pmatrix} 0 & 1 \\ 1 & h_{l-1} \end{pmatrix} \cdots \begin{pmatrix} 0 & 1 \\ 1 & h_0 \end{pmatrix} &= \begin{pmatrix} N_1' - \tau Q & N' - \sigma Q \\ \tau & \sigma \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & h_0 \end{pmatrix} \\ &= \begin{pmatrix} N' - \sigma Q & N_1' - \tau Q + h_0(N' - \sigma Q) \\ \sigma & \tau + \sigma h_0 \end{pmatrix} \\ &=: \begin{pmatrix} N' - \bar{\tau} Q & M' - \bar{\sigma} Q \\ \bar{\tau} & \bar{\sigma} \end{pmatrix} \end{aligned}$$

wobei $N'\bar{\sigma} - M'\bar{\tau} = N'(\tau + \sigma h_0) - M'\sigma = N'\tau - N_1'\sigma = (-1)^l, \quad 0 < \bar{\tau} \le N'$ und $0 < \bar{\sigma} \le N_1' + N'h_0 = M'$, wie es sein soll (vgl. 3.9).

Aus (3.11) folgt, daß sich f^* und $T(T(\dots T(f, M, N)\dots, N_{l-3}, N_{l-2}), N_{l-2}, N_{l-1})$ nur um einen Faktor $x^a y^b$ unterscheiden können, aber da ihre Newtondiagramme beide die Achsen gerade berühren, ist dieser Faktor Eins.

Zu (iv): Folgt direkt aus (iii). ▧

Wir haben nun alle Transformationen T zu einer zusammengefaßt (vgl. (iii)). Es ist aber klar, daß nach der letzten dieser T eine Transformation T_1 folgen muß – wir haben ja auch in (ii) gesehen, daß für die Transformierte f^* tatsächlich der Fall $R = 0$ eintritt. Wir wollen diese Transformation T_1 noch mit T^* zusammenfassen.

Dabei stellt sich die Frage, ob sich das δ in Schritt 3 von `HNE-irred-komplett` mit $L(f^*) = c\left(y - \delta x^Q\right)^e$ auch schon ohne Kenntnis von f^* berechnen läßt. Dies läßt sich bejahen, denn:

Nach Lemma 3.5 gibt es $\quad c, \delta \in k^* : L(f) = c\left(y^{N'} - \delta x^{M'}\right)^e$

$$\begin{aligned}
\Longrightarrow \quad L(f^*) &= L\left(T^*(f, M', N')/x^s y^t\right) \underbrace{=}_{3.12(\text{iii}),\ 3.7(\text{ii})} T^*(L(f), M', N') \;/\; x^s y^t \\
&= c\left(x^{M'N'-\sigma QN'} y^{\sigma N'} - \delta x^{M'N'-\tau QM'} y^{\tau M'}\right)^e \Big/ \; x^s y^t \\
&= c\Big(x^{Q\max\{N'\sigma, M'\tau\}-\sigma QN'} y^{\sigma N' - \min\{N'\sigma, M'\tau\}} \\
&\qquad -\delta x^{Q\max\{N'\sigma, M'\tau\}-\tau QM'} y^{\tau M' - \min\{N'\sigma, M'\tau\}}\Big)^e
\end{aligned}$$

Falls $\quad \max\{N'\sigma, M'\tau\} = N'\sigma \quad (\iff N'\sigma - M'\tau = 1 \iff l$ gerade$), \quad$ dann ist

$$L(f^*) = c\left(y^{N'\sigma - M'\tau} - \delta x^{Q(N'\sigma - M'\tau)}\right)^e = c\left(y - \delta x^Q\right)^e$$

sonst $\quad (\iff l$ ungerade$) \quad$ ist

$$L(f^*) = c\left(x^{Q(M'\tau - N'\sigma)} - \delta y^{M'\tau - N'\sigma}\right)^e = c\left(x^Q - \delta y\right)^e = c(-\delta)^e\left(y - \frac{1}{\delta} x^Q\right)^e$$

Also: Wenn $\quad L(f) = c\left(y^{N'} - \delta x^{M'}\right)^e \quad$ und $\quad L(f^*) = c'\left(y - \delta' x^Q\right)^e \quad$ ist, dann gilt:

$$\begin{array}{ll} \delta' = \delta & \text{falls } l \text{ gerade } (\iff N'\sigma > M'\tau) \\ \delta' = \frac{1}{\delta} & \text{falls } l \text{ ungerade } (\iff N'\sigma < M'\tau) \end{array} \tag{3.12}$$

δ wiederum läßt sich aus f mittels zu (3.6), (3.7) analogen Formeln berechnen:

$$\sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, x^\alpha y^\beta = c\left(y^{N'} - \delta x^{M'}\right)^e = \sum_{i=0}^{e} c \binom{e}{i} y^{N-N'i} (-\delta)^i x^{M'i}$$

daher ist $\quad c = c_{0,N} \quad$ wie gehabt.

Fall 1: $\quad \operatorname{char} k \nmid e. \quad$ Dann folgt $\quad -ce\delta\, y^{N-N'} x^{M'} = c_{M',N-N'}\, y^{N-N'} x^{M'}$

$$\Longrightarrow \qquad \boxed{\delta = -\frac{c_{M',N-N'}}{e\, c_{0,N}}} \tag{3.13}$$

Fall 2: $\quad \operatorname{char} k = p, \quad e = \epsilon\, p^j \quad$ mit $\quad p \nmid \epsilon. \quad$ Dann folgt

$$c\left(y^{N'} - \delta x^{M'}\right)^e = c\left(y^{N'p^j} - \delta^{p^j} x^{M'p^j}\right)^\epsilon \quad \Longrightarrow \quad -\delta^{p^j} \epsilon c = c_{M'p^j,\, N'p^j(\epsilon-1)}$$

$$\Longrightarrow \qquad \boxed{\delta = \sqrt[p^j]{-\frac{c_{M'p^j,\, N-N'p^j}}{\epsilon\, c_{0,N}}}} \tag{3.14}$$

Mit $\delta' \in \{\delta, 1/\delta\}$ wie in (3.12) und s, t, Q wie in Lemma 3.12 definieren wir schließlich für irreduzible $f \in k[\![x,y]\!]$:

Definition 3.8

$$\boxed{\boldsymbol{T_2}(f, \delta', M', N') := T_1\left(\frac{T^*(f, M', N')}{x^s y^t}, \delta', Q\right)}$$

Bemerkung: T_1 ist ein Spezialfall von T_2, denn falls $R = 0$, dann ist
$M = QN,\ l = 0,\ N' = 1,\ M' = Q \quad \Rightarrow \quad \sigma = 1,\ \tau = 0,\ \delta' = \delta$
$\Rightarrow \quad s = MN' - QN = 0, \quad t = 0$

$$\Longrightarrow \quad T_2\left(f, \delta', M', N'\right) = T_1\Big(f\left(x^1 y^0, x^{M'-Q} y^1\right), \delta', Q\Big) = T_1\left(f, \delta, Q\right)$$

Aufgrund dieser Bemerkung kann Schritt 3 von `HNE-irreduzibel` auch kürzer wie folgt geschrieben werden:

Schritt 3: Setze $e \leftarrow \mathrm{ggT}(M, N), \quad M' \leftarrow M/e, \quad N' \leftarrow N/e.$
Führe den euklidischen Algorithmus für M und N aus und setze die Werte von $h_s, \ldots, h_{s+l-1}$ entsprechend (3.8)
sowie $a_{j,i} \leftarrow 0$ für $s \leq j \leq s + l$ und die entsprechenden i
setze $s \leftarrow s + l, \quad q \leftarrow Q, \quad Q$ wie in (3.8).
Berechne σ, τ wie in (3.9) (z. B. mit euklid. Algorithmus angewandt auf M' und N'), berechne c, δ gemäß (3.13) bzw. (3.14).
`falls` $L(f) \neq c\left(y^{N'} - \delta x^{M'}\right)^e$ `dann STOP("f reduzibel!")`
`falls` $N'\sigma > M'\tau$ `dann` $\delta' \leftarrow \delta$ `sonst` $\delta' \leftarrow 1/\delta$
setze $a_{s,Q} \leftarrow \delta', \quad f \leftarrow T_2(f, \delta', M', N')$
`falls` $y \mid f$ `dann` setze $a_{s,j} \leftarrow 0$ für alle $j > q$ und `STOP("Ende")`
`falls` $N = 1$ `dann STOP("Abbruch")`
gehe zurück zu Schritt 2.

Bei Verwendung dieses modifizierten Algorithmus kann man sich bei jedem Durchlauf von Schritt 3 merken, mit welchen Parametern T_2 aufgerufen wurde. Aus diesen läßt sich nämlich die berechnete HNE rekonstruieren. Dieses Vorgehen wird sich als sehr nützlich für den reduziblen Fall herausstellen.

Definition 3.9 Sei $f \in k[\![x,y]\!]$ irreduzibel und der modifizierte Algorithmus `HNE-irreduzibel` möge u-mal den Schritt 3 durchlaufen, dabei werde beim i-ten Durchlauf die Transformation $T_2(\cdot, \delta_i', M_i', N_i')$ ausgeführt. Dann heißt

$$\begin{array}{l} (M_1', N_1', (\delta_1' \downarrow)) \\ \quad (M_2', N_2', (\delta_2' \downarrow)) \\ \qquad\quad \cdots\cdots \\ \qquad (M_u', N_u', (\delta_u' \downarrow)) \\ \qquad\qquad\quad (\texttt{"Ende"}) \end{array} \qquad \text{oder} \qquad \begin{array}{l} (M_1', N_1', (\delta_1' \downarrow)) \\ \quad (M_2', N_2', (\delta_2' \downarrow)) \\ \qquad\quad \cdots\cdots \\ \qquad (M_u', N_u', (\delta_u' \downarrow)) \\ \qquad\qquad\quad (\texttt{"Abbruch"}) \end{array}$$

eine **charakteristische Liste**[4] von f. Jede charakteristische Liste, die nicht mit `"Ende"` aufhört, kann (durch weitere Transformationen von f mit T_1) zu einer unendlich langen Liste fortgesetzt werden, oder zu einer Liste, die mit `"Ende"` abbricht, je nachdem, ob die HNE von f abbricht oder nicht. Eine solche Liste nennen wir eine **Hamburger-Noether-Liste** von f.

Bemerkung: Hamburger-Noether-Liste und -Entwicklung von f lassen sich ineinander umrechnen (insbesondere ist die Hamburger-Noether-Liste von f eindeutig), ebenso kann man aus der charakteristischen Liste von f die HNE von f bis zum Anfang der letzten Zeile berechnen.

Beispiel 1: $k = \mathbb{F}_2, \quad f = y^2 + x^7y + x^8 + x^9$ (vgl. Beispiel 1 auf S. 60)

Schritt 2: $M = 8, \quad N = 2.$ Schritt 3 neu: $e = 2, \quad M' = 4, \quad N' = 1.$
Euklidischer Algorithmus: $M = 4N + 0$
$L(f) = \left(y^{N'} + x^{M'}\right)^e \rightsquigarrow \delta = 1, \quad a_{0,4} = 1.$
$f_1 := T_2(f,1,4,1) = T_1(f,1,4) = y^2 + x^7y + x^9 + x^{11}$ wie gehabt.

Wir merken uns $M_1' := 4, \quad N_1' := 1, \quad \delta_1' := 1$

Schritt 2: $M = 9, \quad N = 2.$ Schritt 3 neu: $e = 1, \quad M' = 9, \quad N' = 2.$

Euklidischer Algorithmus: $\begin{array}{rcl} M & = & 4N+1 \\ N & = & 2\cdot 1 + 0 \end{array} \rightsquigarrow h_0 = 4, \quad Q = 2$

$4\cdot 2 - 1\cdot 9 = -1 \rightsquigarrow \sigma = 4, \quad \tau = 1$
$L(f_1) = y^2 + x^9 = \left(y^{N'} + x^{M'}\right)^e \rightsquigarrow \delta = 1.$
$N'\sigma = 8 < 9 = M'\tau \rightsquigarrow \quad \delta' = 1/\delta = 1, \quad a_{1,2} = 1,$
$\qquad s = MN' - QM\tau = 18 - 18 = 0, \quad t = N'\sigma = 8.$

$$\begin{aligned} f_3 := T_2(f_1,1,9,2) &= T_1\left(\frac{f(x^{2-2}y^1, x^{9-8}y^4)}{y^8}, 1, 2\right) \\ &= T_1\left(x^2 + xy^3 + y + y^3, 1, 2\right) = y + (1+x)\left(y + x^2\right)^3 \end{aligned}$$

Wir merken uns $M_2' := 9, \quad N_2' := 2, \quad \delta_2' := 1$

Schritt 2: $M = 6, \quad N = 1.$ Schritt 3: $e = 1, \quad M' = 6, \quad N' = 1.$
$L(f_3) = y + x^6 = \left(y^{N'} + x^{M'}\right)^e \rightsquigarrow \delta = 1, \quad a_{1,6} = 1$
$f_4 := T_2(f_3,1,6,1) = T_1(f_3,1,6).$

Wir merken uns $M_3' := 6, \quad N_3' := 1, \quad \delta_3' := 1$
und sind wegen $N = 1$ fertig.

Damit lautet die charakteristische Liste von f:

$$\begin{array}{l} (4,1,(1\downarrow)) \\ \quad (9,2,(1\downarrow)) \\ \qquad (6,1,(1\downarrow)) \\ \qquad\quad (\texttt{"Abbruch"}) \end{array} \qquad \text{woraus wir wiederum} \qquad \begin{array}{rcl} z_{-1} & = & z_0^4 + z_0^4 z_1 \\ z_0 & = & z_1^2 + z_1^6 + \ldots \end{array}$$

als HNE von f ablesen können.

[4] Wie sich später herausstellen wird, sind die Kurveninvarianten von f durch diese Liste eindeutig bestimmt, daher der Name.

Beispiel 2: Sei $k = \mathbb{Q}$ und R eine irreduzible ebene algebroide Kurve mit der Hamburger-Noether-Entwicklung

$$
\begin{array}{rcl}
z_{-1} & = & z_0^3 - z_0^5 \qquad\qquad + z_0^6 z_1 \\
z_0 & = & z_1^2 z_2 \\
z_1 & = & z_2^4 z_3 \\
z_2 & = & 2z_3^3 + 3z_3^4 \qquad + z_3^4 z_4 \\
z_3 & = & 7z_4^8 - 5z_4^{11} - 3z_4^{14} + 6z_4^{15}
\end{array}
$$

Dann sind

$$
\begin{array}{l}
(3,1,(1\downarrow)) \\
(5,1,(-1\downarrow)) \\
(187,29,(2\downarrow)) \\
(4,1,(3\downarrow)) \\
(33,8,(7\downarrow)) \\
(11,1,(-5\downarrow)) \\
(\texttt{"Abbruch"})
\end{array}
\qquad \text{und} \qquad
\begin{array}{l}
(3,1,(1\downarrow)) \\
(5,1,(-1\downarrow)) \\
(187,29,(2\downarrow)) \\
(4,1,(3\downarrow)) \\
(33,8,(7\downarrow)) \\
(11,1,(-5\downarrow)) \\
(14,1,(-3\downarrow)) \\
(15,1,(6\downarrow)) \\
(\texttt{"Ende"})
\end{array}
$$

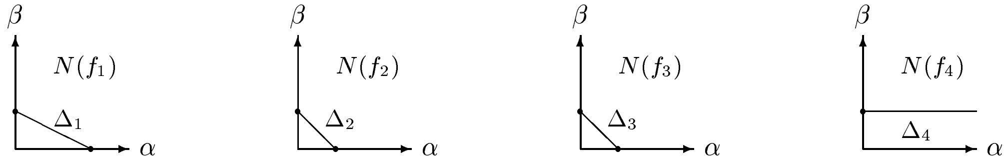
die charakteristische Liste bzw. die Hamburger-Noether-Liste von R.

3.2 Reduzibler Fall

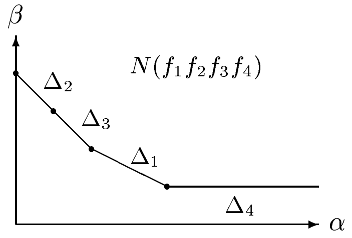
Bis auf weiteres setzen wir ab jetzt voraus, daß k ein algebraisch abgeschlossener Körper ist.

Im irreduziblen Fall haben wir gesehen, daß das Newtonpolygon eine wichtige Rolle bei der Berechnung der Hamburger-Noether-Entwicklung spielt. Natürlich geht die Eigenschaft, daß es eine Halbgerade ist oder aus nur einer Seite besteht, im reduziblen Fall verloren, aber es gilt das folgende Lemma mit zentraler Bedeutung:

Lemma 3.13 *Sei $f = f_1 \cdots f_n \in k[[x,y]]$, f_i irreduzibel, dann erhält man das Newtonpolygon von f, indem man die (verschobenen) Newtonpolygone der f_i (je eine Seite/Halbgerade) in einer solchen Reihenfolge aneinanderhängt, daß als Ergebnis eine konvexe Linie herauskommt.*



Beweis: [BK], Kap. 8.4, Lemma 3



Falls das Newtonpolygon eine oder zwei Halbgeraden enthält, ist die gegebene Potenzreihe f durch x und/oder y teilbar. Die HNEs dieser Zweige sind aber bekannterweise $\quad z_{-1} = 0 \quad$ mit $z_{-1} = [y]_f$ oder $z_{-1} = [x]_f$. Wir können uns also diese Zweige merken, f durch x und/oder y dividieren und ab jetzt immer annehmen:

Generalvoraussetzung: $f \in k[[x,y]]$ sei eine quadratfreie Potenzreihe (keine Einheit und auch nicht Null), die weder durch x noch durch y teilbar ist, k sei ein algebraisch abgeschlossener Körper. Das Newtonpolygon von f bestehe also nur aus Seiten endlicher Länge und Höhe.

Da sich nach dem eben genannten Lemma hinter jeder Seite des Newtonpolygons ein (nicht notwendigerweise irreduzibler) Faktor von f verbirgt, ist unser Ansatz der, eine Seite auszuwählen und in Abhängigkeit von ihrer Länge und Höhe (möglichst) die gleichen Transformationen wie im irreduziblen Fall anzuwenden. Dazu untersuchen wir das Verhalten des Newtonpolygons unter diesen Transformationen.

3.2.1 Die Transformation T_1

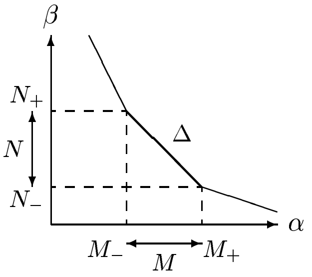
Um den richtigen Parameter δ für T_1 zu ermitteln, haben wir im irreduziblen Fall $L(f)$ zusammen mit Lemma 3.5 benutzt. Wir verallgemeinern beides für beliebige

Potenzreihen f:

Definition 3.10 Für eine Seite Δ aus dem Newtonpolygon von $f = \sum c_{\alpha\beta}\, x^\alpha y^\beta$ definieren wir die quasihomogene Leitform genau wie im irreduziblen Fall:

$$\boldsymbol{L(f)} := L_\Delta(f) := \sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, x^\alpha y^\beta$$

Lemma 3.14 *Es sei Δ eine Seite von $N(f)$ mit Länge M und Höhe N,* $M_- := \min\left\{ \alpha \mid (\alpha,\beta) \in \Delta \right\}$, $\quad N_- := \min\left\{ \beta \mid (\alpha,\beta) \in \Delta \right\}$, $e := \mathrm{ggT}(M,N)$, $\quad M' := M/e \quad$ *und* $\quad N' := N/e$.



Dann gibt es $\quad c, \delta_i \in k^*$ *(die δ_i paarweise verschieden) und $e_i \in \mathbb{N}$, so daß*

$$L_\Delta(f) = c x^{M_-} y^{N_-} \prod_i \left(y^{N'} - \delta_i x^{M'} \right)^{e_i} \quad \text{und} \quad \sum_i e_i = e \tag{3.15}$$

Beweis: $\quad x^{M_-} y^{N_-} \mid L_\Delta(f)$. $\qquad$ Definiere $\quad L' := L_\Delta(f)/x^{M_-} y^{N_-}$.
Dann gilt $\quad x \nmid L'$, $\quad y \nmid L' \quad$ und $\quad N(L')$ besteht aus genau einer Seite der Länge M und der Höhe N. Daher gibt es für jeden irreduziblen Faktor S von L' ein $c, \delta \in k^*$:

$$S = c\left(y^{N'} - \delta x^{M'} \right)$$

(vgl. z. B. Lemma 3.5, denn die quasihomogene Leitform von S ist S).
Nun muß man nur noch alle c sowie die Faktoren mit gleichem δ zusammenfassen. Die Behauptung $\quad \sum e_i = e \quad$ folgt ebenfalls daraus, daß $N(L')$ die Höhe N hat, und aus der gerade bewiesenen Formel für $L_\Delta(f)$. ▧

Wir wählen nun eine Seite Δ aus $N(f)$ aus, deren Höhe N ihre Länge M teilt. Dann ist $M' = M/N$, $N' = 1$ und nach Lemma 3.14 gibt es $c, \delta_i \in k^*$ und $e_i \in \mathbb{N}$, so daß

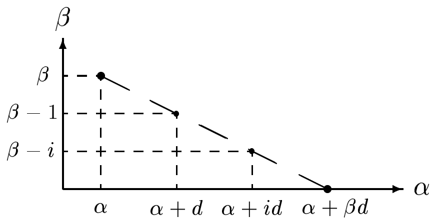
$$L_\Delta(f) = c x^{M_-} y^{N_-} \prod_i \left(y - \delta_i x^{M'} \right)^{e_i} \tag{3.16}$$

Betrachten wir, welche Auswirkungen T_1 für das Newtondiagramm verschiedener Potenzreihen hat:

1) Seien $\delta \in k^*$ und $\alpha, \beta, d \in \mathbb{N}$ beliebig. Dann ist

$$T_1(x^\alpha y^\beta, \delta, d) = x^\alpha \left(y + \delta x^d \right)^\beta = \sum_{i=0}^{\beta} \delta^i \binom{\beta}{i} x^{\alpha+id} y^{\beta-i}$$

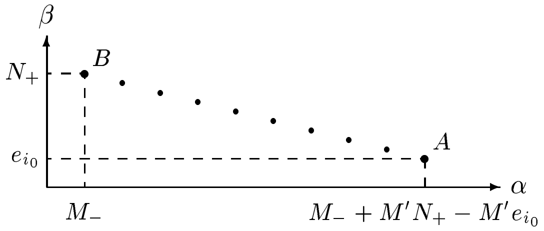
Die Monome von $T_1(x^\alpha y^\beta, \delta, d)$ liegen im Newtondiagramm auf einer Geraden mit Steigung $-1/d$, die Endpunkte (α, β) und $(\alpha + \beta d, 0)$ gehören auf jeden Fall dazu. Ob $D(T_1(x^\alpha y^\beta, \delta, d))$ weitere Punkte hat, hängt von β und der Charakteristik von k ab.



2) Sei δ_{i_0} eines der δ_i aus Lemma 3.14. Dann ist wegen $\sum e_i = e = N$

$$
\begin{aligned}
T_1\left(L_\Delta(f), \delta_{i_0}, M'\right) &\underset{(3.16)}{=} cx^{M_-}\left(y + \delta_{i_0} x^{M'}\right)^{N_-} \prod_i \left(y + (\delta_{i_0} - \delta_i) x^{M'}\right)^{e_i} \\
&= cx^{M_-} y^{e_{i_0}} \left(y + \delta_{i_0} x^{M'}\right)^{N_-} \prod_{i \neq i_0} \left(y + (\delta_{i_0} - \delta_i) x^{M'}\right)^{e_i} \\
&= cx^{M_-} y^{e_{i_0} + N_- + N - e_{i_0}} + \ldots + cx^{M_- + M'N_- + M'(N - e_{i_0})} y^{e_{i_0}} \underbrace{\delta_{i_0}^{N_-} \prod_{i \neq i_0} (\delta_{i_0} - \delta_i)^{e_i}}_{\neq 0} \\
&= \square x^{M_-} y^{N_+} + \ldots + \square x^{M_- + M'N_+ - M'e_{i_0}} y^{e_{i_0}}
\end{aligned}
$$

wobei $\square$ für irgendeine von Null verschiedene Zahl steht.



Die Punkte $A = (M_- + M'N_+ - M'e_{i_0},\, e_{i_0})$ und $B = (M_-, N_+)$ kommen also im Newtondiagramm von $T_1\left(L_\Delta(f), \delta_{i_0}, M'\right)$ vor. Teil 1) hat gezeigt, daß alle Monome dieses Polynoms im Newtondiagramm auf einer Linie zwischen B und $(M_- + M'N_+,\, 0)$ liegen (denn die Linie in 1) hat dieselbe Steigung wie die Seite Δ), und beim Ausmultiplizieren von $T_1\left(L_\Delta(f), \delta_{i_0}, M'\right)$ haben wir gesehen, daß alle Monome durch $y^{e_{i_0}}$ teilbar sind.

Es gilt also: Alle Punkte von $D(T_1\left(L_\Delta(f), \delta_{i_0}, M'\right))$ liegen auf der Verbindungslinie von A und B, wobei $A, B \in D(T_1\left(L_\Delta(f), \delta_{i_0}, M'\right))$.

3) Schließlich betrachten wir $T_1(f, \delta_{i_0}, M')$ selbst.

- Aus 1) folgt: Der Teil von $N(f)$ oberhalb des Punktes B wird nicht verändert, weil die Steigung aller Seiten in diesem Teil steiler als $-1/M'$ ist.
- Aus 2): Δ wird transformiert in eine Seite Δ' gleicher Steigung $(-1/M')$ und mit gleichem Anfangspunkt B, aber mit neuem Endpunkt A; dieser befindet sich im Abstand e_{i_0} von der Abszisse.
- Außerdem erhält man eine gewisse Anzahl (> 0) von Seiten mit Steigung echt größer als $-1/M'$ – es kann auch eine horizontale Halbgerade darunter sein (folgt wieder aus 1).

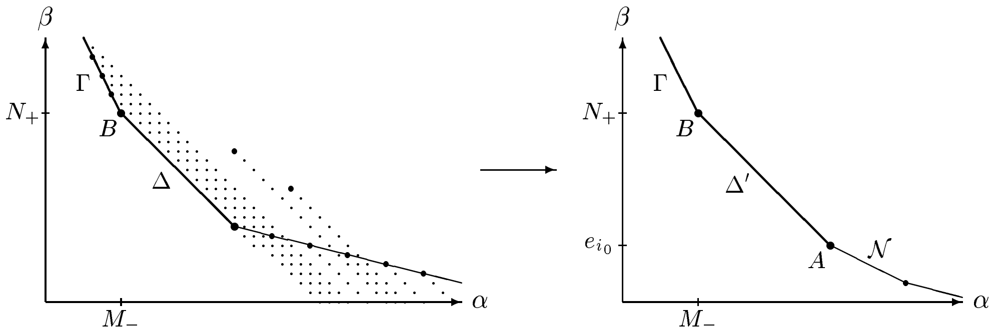


Abbildung 3.2: Transformationsverhalten des Newtonpolygons unter T_1

3.2.2 Die Transformation T_2

Sei wieder Δ eine Seite aus $N(f)$ mit Höhe N und Länge M, aber diesmal sei $M = NQ + R$ für ein $Q > 0$ und $0 < R < N$.
Wir betrachten zuerst die Transformation $U(f, M, N) = f\left(y, xy^Q\right)$:

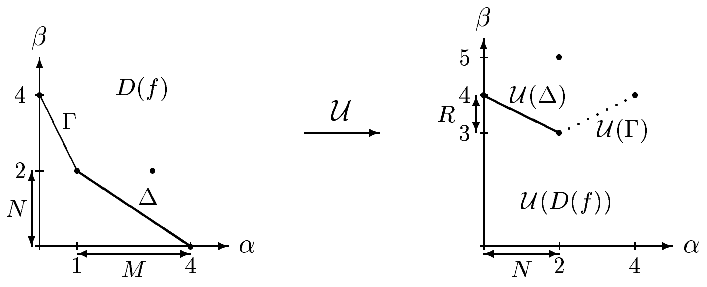
Das Newtondiagramm von $U(f, M, N)$ erhält man aus $D(f)$ durch Anwendung der affinen Transformation

$$\mathcal{U} : \mathbb{Z}_+^2 \longrightarrow \mathbb{Z}_+^2\,, \quad \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \mapsto \begin{pmatrix} 0 & 1\\ 1 & Q\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha + Q\beta\end{pmatrix}$$

Anders als im irreduziblen Fall ist aber $N(U(f, M, N))$ im allgemeinen *nicht* gleich $\mathcal{U}(N(f))$ (Halbgeraden nicht mitgerechnet), denn es kann Seiten von $N(f)$ geben, die von $\mathcal{U}$ in Seiten mit positiver Steigung überführt werden und damit aus dem Newtonpolygon wegfallen. Darum müssen wir die Definition von T und T_2 für den reduziblen Fall verallgemeinern.

Beispiel: $f = (y^2 + x)(y^2 + x^3)$: Wir wählen Δ so, daß $M = 3, N = 2$, d. h. Δ das Newtonpolygon des zweiten Faktors ist.

$$U(f, 3, 2) = f(y, xy) = (x^2y^2 + y)(x^2y^2 + y^3) = y^3(x^2y + 1)(x^2 + y)$$



Das richtige Monom x^sy^t, durch das zu dividieren ist, läßt sich insbesondere bei T_2 nur schwer a priori bestimmen, dagegen ist es a posteriori ganz leicht. Wir definieren T neu:

Definition 3.11 Sei $f \in k[[x,y]]$ und Δ eine Seite aus $N(f)$ mit Länge M und Höhe $N < M$. Sei ferner $t \in \mathbb{N}$ maximal gewählt mit der Eigenschaft $y^t \mid U(f,M,N)$. Dann definieren wir

$$\boxed{\boldsymbol{T}(f,M,N)(x,y) := \frac{U(f,M,N)(x,y)}{y^t} = \frac{f\left(y, xy^Q\right)}{y^t}}, \quad Q \text{ wie üblich.}$$

Man beachte, daß diese Definition äquivalent ist zu Definition 3.5, falls f irreduzibel ist. $x \nmid U(f,M,N)$, denn $\mathcal{U}(\alpha,0) = (0,\alpha)$.

Bemerkung: Definieren wir wieder

$$\mathcal{T} : \mathbb{Z}^2 \longrightarrow \mathbb{Z}^2, \quad \begin{pmatrix}\alpha\\ \beta\end{pmatrix} \mapsto \begin{pmatrix}0 & 1\\ 1 & Q\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} - \begin{pmatrix}0\\ t\end{pmatrix}$$

so sieht man sofort, daß $\mathcal{T}(\Delta)$ eine Seite der Länge N und der Höhe R ist (man betrachte die Bilder der Endpunkte von Δ), so wie im irreduziblen Fall (R wie immer der Rest der Division).

Wir können daher wiederum alle Transformationen T und die erste darauffolgende Transformation T_1 zu einer Transformation T_2 zusammenfassen, indem wir den euklidischen Algorithmus auf M und N anwenden. Seien dazu N_i die Reste der einzelnen Divisionen (vgl. 3.8).

Nach Gleichung (3.11) ist $T^*(\cdot,M',N')$ die Komposition aller Transformationen $U(\cdot,N_{i-1},N_i)$. Daher gilt

Lemma 3.15 *Sei $f \in k[[x,y]]$, Δ eine Seite aus $N(f)$ mit Länge M und Höhe $N \leq M$, und es gelte (3.8) und (3.9). Wir wählen*

$$s,t \in \mathbb{N} \quad \textit{maximal, so daß gilt:} \quad x^s y^t \mid T^*(f,M',N')$$

und definieren

$$f^* := \frac{T^*(f,M',N')}{x^s y^t} = \frac{f\left(x^{N'-\tau Q}y^{\tau},\ x^{M'-\sigma Q}y^{\sigma}\right)}{x^s y^t}$$

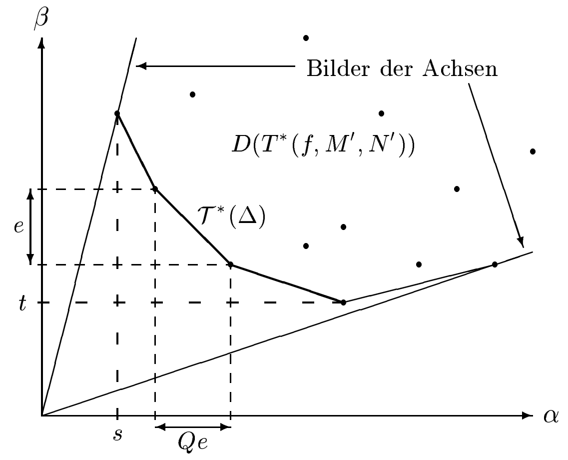
Dann gilt:

(i) $f^* = T(T(\ldots T(f,M,N)\ldots,N_{l-3},N_{l-2}),N_{l-2},N_{l-1})$

(ii) $N(f^*)$ *besitzt eine Seite der Länge Qe und der Höhe* $e = \mathrm{ggT}(M,N)$.

Beweis: (i) folgt analog zum Beweis von 3.12 (iii) aus (3.11); (ii) folgt aus (i) und der vorangegangenen Bemerkung – die gesuchte Seite ist $\mathcal{T}^*(\Delta)$ verschoben um $(-s,-t)$.

Diese Seite (mit Länge Qe und Höhe e) wählen wir jetzt aus und bezeichnen sie mit Δ_1. Dann ist klar, daß als nächstes $T_1(\cdot,\delta',Q)$ für ein geeignetes δ' auszuführen



ist. Um die möglichen δ' aus f zu berechnen, gehen wir analog zum irreduziblen Fall vor:

Nach Lemma 3.14 gibt es geeignete $M_-, N_-, c, \delta_i, e_i$, so daß

$$L_\Delta(f) = cx^{M_-}y^{N_-}\prod_i \left(y^{N'} - \delta_i x^{M'}\right)^{e_i}, \qquad \sum_i e_i = e$$

Lemma 3.16 *Falls $N'\sigma > M'\tau$ in (3.9) ($\iff$ l gerade), dann ist*

$$L_{\Delta_1}(f^*) = c^* x^{M^*_-} y^{N^*_-} \prod_i \left(y - \delta_i x^Q\right)^{e_i}$$

für ein geeignetes Monom $c^* x^{M^*_-} y^{N^*_-} \in k[x,y]$,

sonst ($N'\sigma < M'\tau$ in (3.9) $\iff$ l ungerade) ist

$$L_{\Delta_1}(f^*) = c^* x^{M^*_-} y^{N^*_-} \prod_i \left(y - \frac{1}{\delta_i}x^Q\right)^{e_i}$$

Beweis: Wir definieren

$$M^1_- := M_-(N' - \tau Q) + N_-(M' - \sigma Q) - s, \qquad N^1_- := M_-\tau + N_-\sigma - t$$

Fall 1: $N'\sigma > M'\tau$. Dann folgt

$$\begin{aligned} L_{\Delta_1}(f^*) &= \frac{T^*\left(L_\Delta(f), M', N'\right)}{x^s y^t} \\ &= cx^{M^1_-}y^{N^1_-}\prod_i \left[x^{M'N' - QN'\sigma} y^{M'\tau}\left(y^{N'\sigma - M'\tau} - \delta_i x^{Q(N'\sigma - M'\tau)}\right)\right]^{e_i} \\ &= cx^{M^1_- + M'N'e - QN'\sigma e} y^{N^1_- + M'\tau e} \cdot \prod_i \left(y - \delta_i x^Q\right)^{e_i} \end{aligned}$$

Fall 2: $N'\sigma < M'\tau$. Wir erhalten

$$\begin{aligned} L_{\Delta_1}(f^*) &= cx^{M^1_-}y^{N^1_-}\prod_i \left[x^{M'N' - QM'\tau} y^{N'\sigma}\left(x^{Q(M'\tau - N'\sigma)} - \delta_i y^{M'\tau - N'\sigma}\right)\right]^{e_i} \\ &= \left(c\prod_i(-\delta_i)^{e_i}\right) x^{M^1_- + M'N'e - QM'\tau e} y^{N^1_- + N'\sigma e} \cdot \prod_i \left(y - \frac{1}{\delta_i}x^Q\right)^{e_i} \end{aligned}$$ ▧

Die möglichen δ' sind also aus $\{\delta_1,\ldots,\delta_n\}$ oder $\{1/\delta_1,\ldots,1/\delta_n\}$, je nachdem. Nun können wir T_2 für den reduziblen Fall neu definieren:

Definition 3.12 Sei $f \in k[[x,y]]$ und Δ eine Seite aus $N(f)$ mit Länge M und Höhe $N \le M$. Wir definieren Q,σ,τ durch (3.8) und (3.9) und wählen $s,t \in \mathbb{N}$ maximal mit der Eigenschaft $x^s y^t \mid T^*(f,M',N')$. Schließlich wählen wir noch ein δ_{i_0} aus den δ_i von Lemma 3.14 aus und definieren $\delta'_{i_0} := \delta_{i_0}$ falls $N'\sigma > M'\tau$, sonst $\delta'_{i_0} := 1/\delta_{i_0}$. Dann ist

$$\boxed{\boldsymbol{T_2}(f,\delta'_{i_0},M',N') := T_1\left(\frac{T^*(f,M',N')}{x^s y^t},\delta'_{i_0},Q\right)}$$

Wiederum ist T_1 ein Spezialfall von T_2, denn wie bei der Bemerkung zur Definition von T_2 im irreduziblen Fall gesehen, ist $T^*(f,M',1) = f$, und $x \nmid f$, $y \nmid f$ nach Voraussetzung.

Wir untersuchen jetzt, wie man mit dieser Transformation die Hamburger-Noether-Entwicklung von zunächst *einem* Zweig von $k[[x,y]]\,/\,\langle f\rangle$ berechnen kann. Es sei

$$f = \prod_{i=1}^{n} f_i, \qquad f_i \quad \text{irreduzibel}$$

und wir suchen uns einen Faktor aus; dieser sei f_1. O.B.d.A. sei $[x]_{f_1} \in k[[x,y]]\,/\,\langle f_1\rangle$ ein transversaler Parameter (sonst vertausche x und y in f).
Das Newtonpolygon von f_1 besteht aus einer Seite, diese habe die Steigung $-\frac{M'}{N'}$.
Wegen Lemma 3.13 hat $N(f)$ eine Seite Δ mit der gleichen Steigung.

$$\exists\, I \subseteq \{1,\ldots,n\}: \quad \Delta = N\left(\prod_{i\in I} f_i\right) \qquad \text{Setze} \quad f^* := \prod_{i\in I} f_i$$

$$\exists\, c,c',M_-,N_-,\delta_i: \quad L_\Delta(f) = cx^{M_-}y^{N_-}L_\Delta(f^*),\ L_\Delta(f^*) = c'\prod_{i=1}^{m}\left(y^{N'}-\delta_i x^{M'}\right)^{e_i}$$

$$L_\Delta(f^*) = \prod_{i\in I} L(f_i) \quad \Longrightarrow \quad \exists\, i_0 \in \mathbb{N}, d \le e_{i_0}, c'' \in k^*: L(f_1) = c''\left(y^{N'}-\delta_{i_0}x^{M'}\right)^{d}$$

Das δ_{i_0} von $L(f_1)$ befindet sich folglich unter den δ von $L_\Delta(f)$.

Lemma 3.17
$$T_2\left(\prod_{i=1}^{n} f_i,\ \delta'_{i_0},M',N'\right) = \prod_{i=1}^{n} T_2(f_i,\ \delta'_{i_0},M',N')$$

Beweis: Offensichtlich ist
$$T^*\left(\prod_{i=1}^{n} f_i,\ M',N'\right) = \prod_{i=1}^{n} T^*(f_i,\ M',N')$$

und
$$T_1\left(\prod_{i=1}^{n} g_i,\ \delta'_{i_0},Q\right) = \prod_{i=1}^{n} T_1(g_i,\ \delta'_{i_0},Q)$$

für beliebige g_i.

Weil x und y weder $\dfrac{T^*\left(\prod\limits_{i=1}^{n} f_i,\, M', N'\right)}{x^s y^t}$ noch $\prod\limits_{i=1}^{n} \dfrac{T^*(f_i,\, M', N')}{x^{s_i} y^{t_i}}$ teilen

(s, t, s_i, t_i gemäß der Definition von T_2 gewählt), muß auch $x^s y^t = \prod\limits_{i=1}^{n} x^{s_i} y^{t_i}$ sein. ▧

Im allgemeinen können wir f_1 nicht berechnen, sonst wäre der Algorithmus für den reduziblen Fall ganz einfach: Faktorisiere f und wende `HNE-irreduzibel` auf alle Faktoren an. Wegen obigem Lemma genügt es aber, T_2 auf f anzuwenden, dann erhalten wir ein Produkt, von dem ein Faktor $T_2(f_1)$ ist. Wir haben nur noch herauszufinden, wie das Newtonpolygon von $T_2(f_1)$ aussieht (wenn wir nur $N(T_2(f))$ kennen), dann geht der Rest wie bei `HNE-irreduzibel`.

Die Seite, aus der $N\left(\frac{T^*(f_1,M',N')}{x^{s_1}y^{t_1}}\right)$ besteht, ist ein Teil der Seite Δ_1, und nach Lemma 3.16 gilt für die richtig gewählten $\delta_i' \in \{\delta_i, 1/\delta_i\}$:

$$L_{\Delta_1}\left(\frac{T^*(f, M', N')}{x^s y^t}\right) = c^* x^{M_-^*} y^{N_-^*} \prod_{i=1}^{m} \left(y - \delta_i' x^Q\right)^{e_i}$$

Es bezeichne
$$f^\star := \frac{T^*(f, M', N')}{x^s y^t}$$

Aus Teil 3 der Untersuchungen von T_1 folgt, daß $N\left(T_1\left(f^\star, \delta_{i_0}', Q\right)\right) = N\left(T_2\left(f, \delta_{i_0}', M', N'\right)\right)$ ein Teilpolygon $\mathcal{N}$ der Höhe e_{i_0} hat (evtl. mit Halbgeraden), dessen Seiten eine Steigung echt größer als $-1/Q$ haben, während alle anderen Seiten von $N\left(T_1\left(f^\star, \delta_{i_0}', Q\right)\right)$ steiler sind (vgl. Abbildung 3.2 auf S. 74: $\mathcal{N}$ ist der Teil rechts von A).

Andererseits ist nach Lemma 3.6 (i) $N\left(T_1\left(\frac{T^*(f_1,M',N')}{x^{s_1}y^{t_1}}\right)\right)$ eine Seite mit Steigung echt größer als $-1/Q$ oder eine Halbgerade. Wir finden das Newtonpolygon von $T_2(f_1)$ also im unteren Teilpolygon $\mathcal{N}$. Diese Information genügt uns, denn falls f_j ein Faktor von f ist mit

$$L(f_j) = c_j \left(y^{N'} - \delta_{i_1} x^{M'}\right)^{d'}$$

und $\delta_{i_1} \neq \delta_{i_0}$, dann ist

$$T_2(L(f_j)) = c_j \left(y - (\delta_{i_1} - \delta_{i_0}) x^Q\right)^{d'} = L(T_2(f_j))$$

Also taucht $N(T_2(f_j))$ nicht in $\mathcal{N}$ auf. Wenn $N(f_j)$ gar nicht zu Δ gehört, gehört $N(T_2(f_j))$ erst recht nicht zu $\mathcal{N}$ (vgl. Eigenschaften 1 & 3 von T_1).

Wir können darum eine beliebige Seite (oder Halbgerade) aus diesem unteren Teilpolygon auswählen, f^{**} den Faktor von $T_2(f)$ sein lassen, dessen Newtonpolygon mit der gewählten Seite identisch ist, und o.B.d.A. annehmen, daß $T_2(f_1)$ ein Teiler von f^{**} ist. Denn wir haben eben bewiesen:

Lemma 3.18 *Für $1 \leq j \leq n$ ist genau dann das Newtonpolygon von $T_2\left(f_j, \delta'_{i_0}, M', N'\right)$ eine Seite von $\mathcal{N}$, wenn es c_j und d' gibt, so daß $L(f_j) = c_j\left(y^{N'} - \delta_{i_0} x^{M'}\right)^{d'}$ ist.*

3.2.3 Algorithmen

Aus dem Lemma leiten wir einen (noch vage formulierten) Algorithmus zur Berechnung eines der Zweige einer algebroiden Kurve her:

Algorithmus `HNE-1-Zweig`

Eingabe: $f \in k[[x,y]]$ quadratfrei, wobei k ein algebraisch abgeschlossener Körper sein soll.

Ausgabe: Die charakteristische Liste eines Zweiges von f, für den $[x]$ transversal ist.

Schritt 1: **`falls`** $y \mid f$ **`dann`** `STOP` (markiere Liste mit **`"Ende"`**)
`sonst` wähle eine Seite Δ aus $N(f)$ aus, $N \leftarrow \text{Höhe}(\Delta)$, $M \leftarrow \text{Länge}(\Delta)$, für die gilt: $N \leq M$
(wenn es kein Δ gibt: `STOP`(„kein Zweig mit transversalem $[x]$ vorhanden!"))
`falls` $N = 1$ **`dann`** ist der entsprechende Zweig glatt; berechne δ mit $L_\Delta(f) = cx^{M_-}y^{N_-}\left(y - \delta x^M\right)$ und füge $(M, 1, (\delta, \text{"}\mathtt{Abbruch}\text{"}))$ zur Liste hinzu, `STOP`.

Schritt 2: Berechne M', N', e, e_i und δ_i wie in Lemma 3.14, d. h.

$$L_\Delta(f) = cx^{M_-}y^{N_-}\prod_{i=1}^{m}\left(y^{N'} - \delta_i x^{M'}\right)^{e_i}$$

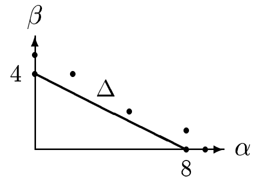
Wähle ein $1 \leq i_0 \leq m$ beliebig, berechne δ'_{i_0}.

Schritt 3: $f \leftarrow T_2(f, \delta'_{i_0}, M', N')$, füge $(M', N', (\delta'_{i_0}, \rightarrow))$ zur Liste hinzu.

Schritt 4: Gehe zurück zu Schritt 1, berücksichtige aber nur noch den Teil des Newtonpolygons, dessen Abstand zur Abszisse kleiner oder gleich e_{i_0} ist.

Bemerkung: Falls f irreduzibel ist, ist `HNE-1-Zweig` äquivalent zu `HNE-irreduzibel` in der modifizierten Form, und zwar weil $e_{i_0} = e = \text{ggT}(M, N)$ ist und wegen Lemma 3.12 (ii).
Je nachdem, wie wir Δ und i_0 in den einzelnen Schritten wählen, können wir die HNE jedes irreduziblen Faktors mit transversalem $[x]$ mit diesem Algorithmus berechnen. Die HNEs der übrigen Faktoren erhalten wir, indem wir am Anfang x und y vertauschen.
Beweisen werden wir erst den endgültigen Algorithmus; in der Hinführung zu `HNE-1-Zweig` wurde außerdem auch schon alles wesentliche gezeigt.

Beispiel: $k = \overline{\mathbb{F}_2}, \quad f := y^4 + x^8 + x^5y^2 + x^9 + y^5 + x^2y^4 + x^8y$



Schritt 1: $N(f)$ hat nur eine Seite, daher ist die Wahl von Δ eindeutig: $M = 8,\ N = 4$.

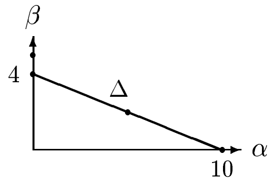
Schritt 2: $M' = 2,\ N' = 1, \quad L_\Delta(f) = y^4 + x^8 = \big(y + x^2\big)^4$.
Wir haben nur $\delta_1 = 1,\ e_1 = 4$ zur Auswahl.

Schritt 3: $T_2(f, \delta_1', 2, 1) = T_1(f, \delta_1, 2) = f(x, y + x^2)$

$$\begin{aligned} &= y^4 + x^8 + x^8 + x^5y^2 + x^9 + x^9 + (y^4 + x^8)(y + x^2) + x^2y^4 + x^{10} + x^8y + x^{10} \\ &= y^4 + x^5y^2 + x^{10} + y^5 =: f_1 \end{aligned}$$

Wir merken uns $(2, 1, (1, \rightarrow))$ als erstes Element der Liste.

Schritt 4: Weil $N(f_1)$ die Gesamthöhe $4 = e_1$ hat, wird auch bei der nächsten Wiederholung das ganze Newtonpolygon von f_1 berücksichtigt.



Schritt 1: Auch $N(f_1)$ hat nur eine Seite Δ;
$M = 10,\ N = 4$.

Schritt 2: $M' = 5,\ N' = 2, \quad L_\Delta(f_1) = y^4 + x^5y^2 + x^{10}$.

Aus Lemma 3.14 folgt, daß sich dieses Produkt noch weiter zerlegen lassen muß; wir machen den Ansatz

$$y^4 + x^5y^2 + x^{10} \overset{!}{=} (y^2 + \delta_1x^5)(y^2 + \delta_2x^5) = y^4 + (\delta_1 + \delta_2)x^5y^2 + \delta_1\delta_2x^{10}$$

$$\Longrightarrow \quad \delta_2 = 1 + \delta_1 \quad \Longrightarrow \quad \delta_1 + \delta_1^2 = \delta_1\delta_2 = 1$$

Wir stellen fest, daß $\delta_1, \delta_2 \notin \mathbb{F}_2$ sind, obwohl $f \in \mathbb{F}_2[\![x, y]\!]$ ist. Die Voraussetzung, daß k vollkommen ist, reicht im reduziblen Fall also nicht aus (vgl. Satz 3.10).

Weil $\delta_1 \neq \delta_2$ ist, erhalten wir $e_1 = 1 = e_2$. Wir wählen $i_0 := 1$.
Wegen $5 = 2 \cdot 2 + 1, \quad 2 = 2 \cdot 1$ ist

$$l = 1 \quad \text{und} \quad T^*(f_1, 5, 2) = U(f_1, 10, 4) = f_1(y, xy^2)$$

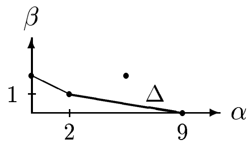
$$\Longrightarrow \quad \delta_1' = \frac{1}{\delta_1} = \frac{\delta_2}{\delta_1\delta_2} = \delta_2$$

Schritt 3: $T^*(f_1, 5, 2) = x^4y^8 + y^5x^2y^4 + y^{10} + x^5y^{10} \quad \Longrightarrow \quad s = 0,\ t = 8$

$$\begin{aligned} \Longrightarrow \quad T_2(f_1, \delta_1', 5, 2) &= T_1(x^4 + x^2y + y^2 + x^5y^2, \delta_2, 2) \\ &= x^4(\underbrace{1 + \delta_2 + \delta_2^2}_{\delta_1 + 1 + \delta_1^2 = 0}) + x^2y + y^2 + x^5y^2 + \delta_2^2x^9 \\ &= y^2 + x^2y + \delta_2^2x^9 + x^5y^2 =: f_2 \end{aligned}$$

Wir merken uns $(5, 2, (\delta_2, \rightarrow))$ als zweites Element der Liste.

Schritt 1: Wegen $e_1 = 1$ berücksichtigen wir nur die unterste Seite von $N(f_2)$. Also ist $M = 7,\ N = 1$.



$$L_\Delta(f_2) = x^2y + \delta_2^2x^9 = x^2\left(y + \delta_2^2x^7\right)$$

Das letzte Element der charakteristischen Liste lautet folglich $(7, 1, (\delta_2^2,$ `"Abbruch"`$))$ und `HNE-1-Zweig` terminiert an dieser Stelle.

Erstes Ergebnis: f besitzt in $\overline{\mathbb{F}_2}[[x,y]]$ einen irreduziblen Faktor, dessen charakteristische Liste

$$\begin{array}{l} (2,1,(1\downarrow)) \\ \quad (5,2,(\delta_2\downarrow)) \\ \qquad (7,1,(\delta_2^2\downarrow)) \\ \qquad\quad (\texttt{"Abbruch"}) \end{array} \quad \text{lautet, d. h.} \quad \begin{array}{lcl} [y] = z_{-1} & = & z_0^2 + z_0^2z_1 \\ [x] = z_0 & = & \delta_2z_1^2 + \delta_2^2z_1^7 + \ldots \end{array}$$

ist der Anfang der HNE. Dabei ist $\delta_2 \in \overline{\mathbb{F}_2}$ eine Nullstelle von $x^2 + x + 1$.

Beim zweiten Durchlauf der Schleife hätten wir natürlich genausogut auch $i_0 := 2$ statt $i_0 := 1$ wählen können. Dies führt zur HNE eines anderen Faktors von f. Will man später die HNEs aller Zweige berechnen, liegt es nahe, den Algorithmus an dieser Stelle verzweigen zu lassen, denn die Rechnung läuft für beide Zweige im ersten Schleifendurchlauf gleich. So brauchen die Transformationen bis zu der Stelle der Verzweigung nur einmal ausgeführt zu werden.

Wir berechnen also die HNE des zweiten Zweiges unter Verwendung der Zwischenergebnisse für den ersten Zweig, d. h. gegeben sind

$$f_1 = y^4 + x^5y^2 + x^{10} + y^5, \quad L_\Delta(f_1) = (y^2 + \delta_1x^5)(y^2 + \delta_2x^5)$$

und $(2, 1, (1, \rightarrow))$ als das erste Element der zu berechnenden Liste. Ferner ist $\delta_2' = 1/\delta_2 = \delta_1$.

Schritt 3:
$$\begin{aligned} T_2(f_1, \delta_2', 5, 2) &= T_1(x^4 + x^2y + y^2 + x^5y^2, \delta_1, 2) \\ &= x^4\underbrace{(1 + \delta_1 + \delta_1^2)}_{0} + x^2y + y^2 + x^5y^2 + \delta_1^2x^9 \\ &= y^2 + x^2y + \delta_1^2x^9 + x^5y^2 =: f_2' \end{aligned}$$

Wir merken uns $(5, 2, (\delta_1, \rightarrow))$ als zweites Element der neuen Liste.

Schritt 1: Auch e_2 ist gleich 1, weswegen wieder nur die unterste Seite von $N(f_2')$ berücksichtigt wird. Es gilt sogar

$$D(f_2') = D(f_2) \ \Rightarrow\ N(f_2') = N(f_2), \quad M = 7,\ N = 1$$

$$L_\Delta(f_2') = x^2y + \delta_1^2x^9 = x^2\left(y + \delta_1^2x^7\right)$$

Endergebnis: f besitzt in $\overline{\mathbb{F}_2}[[x,y]]$ einen weiteren irreduziblen Faktor, dessen charakteristische Liste

$$\begin{array}{l} (2,1,(1\downarrow)) \\ \quad (5,2,(\delta_1\downarrow)) \\ \qquad (7,1,(\delta_1^2\downarrow)) \\ \qquad\quad (\texttt{"Abbruch"}) \end{array} \qquad \text{lautet, d. h.} \qquad \begin{array}{rcl} [y]=z_{-1} & = & z_0^2+z_0^2z_1 \\ [x]=z_0 & = & \delta_1 z_1^2+\delta_1^2 z_1^7+\dots \end{array}$$

ist der Anfang der HNE des zweiten Zweiges. Der Verzweigung bei der Berechnung folgend ordnen wir die charakteristischen Listen der Zweige in einem Baum an:

$$\begin{array}{ccc} & \big(2,1,(1\downarrow)\big) & \\ & \big(5,2,(\delta_1\swarrow),(\delta_2\searrow)\big) & \\ \big(7,1,(\delta_1^2\swarrow)\big) & & \big(7,1,(\delta_2^2\searrow)\big) \\ (\texttt{"Abbruch"}) & & (\texttt{"Abbruch"}) \end{array}$$

Dabei sind $\delta_1,\delta_2\in\overline{\mathbb{F}_2}$ die beiden Nullstellen von x^2+x+1. Wir haben bereits alle Zweige der Kurve berechnet, denn an den Hamburger-Noether-Entwicklungen ist abzulesen, daß beide Zweige die Multiplizität 2 haben. Also hat das Newtonpolygon des Produktes der zugehörigen irreduziblen Faktoren die Höhe 4 (Lemma 3.1), genauso wie auch $N(f)$. Weitere irreduzible Faktoren gibt es daher nicht.

Der eben konstruierte Baum enthält also alle Informationen über die Kurve, um für jeden Zweig mit transversalem Parameter $[x]$ sofort die Hamburger-Noether-Entwicklung ablesen zu können. Wir nennen diesen Baum einen **Auflösungsbaum** von f.

Definition 3.13 Sei $f\in k[[x,y]]$ quadratfrei. Ein **Auflösungsbaum** von f ist ein Baum mit Listen der Form

$$\Big(\dots\big(M',N',\,(\delta\downarrow),(\delta\downarrow),\dots,(\delta\downarrow)\big)\;\dots\dots\;\big(M',N',\,(\delta\downarrow),\dots,(\delta\downarrow)\big)\dots\Big) \qquad (3.17)$$

als Knoten und ("**Abbruch**") oder ("**Ende**") als Blätter, so daß gilt: Für jeden Zweig $k[[x,y]]\,/\,\langle f_i\rangle$ von $k[[x,y]]\,/\,\langle f\rangle$ mit $[x]_{f_i}$ transversal gibt es genau ein Blatt, so daß man beim Durchlaufen des Baumes von der Wurzel zu diesem Blatt eine charakteristische Liste von f_i erhält. Umgekehrt soll natürlich auch jede auf diese Weise erhaltene Liste einem Zweig der Kurve zugeordnet werden können.
In (3.17) stehen dabei M',N' für natürliche Zahlen und δ für (von Null verschiedene) Elemente von k.

Bemerkung:

- Um sämtliche HNEs einer reduziblen Kurve darzustellen, benötigt man folglich zwei Auflösungsbäume: einen von $f(x,y)$ und einen von $f(y,x)$. Dabei kann man bei $f(y,x)$ den zu denjenigen Zweigen gehörenden Teilbaum weglassen, für die beide Parameter $[x]$ und $[y]$ transversal sind.

- Wie im irreduziblen Fall kann eine charakteristische Liste, die aus dem Auflösungsbaum von f gewonnen wurde, zu einer Hamburger-Noether-Liste verlängert werden. Hierbei wird immer nur die unterste Seite von $N(f')$ (die die Höhe 1 hat) berücksichtigt und so lange die Transformation T_1 auf f' angewandt, bis $N(f')$ keinen Schnittpunkt mehr mit der Abszisse hat. Rybowicz schlägt aus diesem Grund vor, den Wert der Transformierten zum Zeitpunkt des Abbruchs der Berechnung in den Auflösungsbaum zu übernehmen, um ggf. die Berechnung der HNE fortsetzen zu können. Das bedeutet, man würde in der Definition die Blätter (`"Abbruch"`) durch (f') ersetzen, $f' \in k[[x,y]]$.

Bevor wir den Algorithmus formulieren, der die Hamburger-Noether-Entwicklungen aller Zweige berechnet, gehen wir noch auf die Berechnung der δ_i in Schritt 2 von `HNE-1-Zweig` ein:

Definition 3.14 Sei $f = \sum c_{\alpha\beta}\, x^\alpha y^\beta \in k[[x,y]]$ und Δ eine Seite von $N(f)$ mit Länge M und Höhe N. Wie üblich bezeichne N_- den y-Wert des unteren Endpunktes von Δ und $N' := N/\mathrm{ggT}(M,N)$. Dann heißt

$$\boldsymbol{\phi_\Delta}(\lambda) := \sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, \lambda^{\frac{\beta-N_-}{N'}} \in k\left[\lambda^{1/N'}\right]$$

das **charakteristische Polynom** von Δ.

Lemma 3.19 *Mit den Voraussetzungen der Definition gilt:*

(i) Das charakteristische Polynom verdient seinen Namen (in dem Sinn, daß $\phi_\Delta(\lambda) \in k[\lambda]$ ist).

(ii) Sind $\delta_1, \ldots, \delta_n$ die Nullstellen von ϕ_Δ mit Vielfachheit $e_1, \ldots, e_n$, dann erfüllen δ_i, e_i die Gleichung (3.15) in Lemma 3.14.

Damit ist das Problem der Berechnung der δ_i, e_i zurückgeführt auf das Faktorisieren univariater Polynome.

Beweis: Nach Lemma 3.14 existieren $c, \delta_i \in k^*$ und $e_i \in \mathbb{N}$, so daß (3.15) gilt. Daraus folgt

$$c\,\lambda^{\frac{N_-}{N'}} \prod_{i=1}^{n} (\lambda - \delta_i)^{e_i} \underset{(3.15)}{=} L_\Delta(f)\left(1, \lambda^{\frac{1}{N'}}\right) = \sum_{(\alpha,\beta)\in\Delta} c_{\alpha\beta}\, \lambda^{\frac{\beta}{N'}} = \lambda^{\frac{N_-}{N'}}\, \phi_\Delta(\lambda)$$

$$\Longrightarrow \qquad \phi_\Delta(\lambda) = c \prod_{i=1}^{n} (\lambda - \delta_i)^{e_i} \in k[\lambda]$$

Der nun folgende Algorithmus benutzt zwei Funktionen, die sich gegenseitig aufrufen. Die erste berechnet alle Teilbäume, die zu den Seiten des Newtonpolygons mit der Eigenschaft (Abstand von der Abszisse $\leq$ eine gegebene Zahl) gehören. Die

zweite berechnet den zu einer bestimmten Seite gehörenden Teilbaum des Auflösungsbaumes.
Falls L_1 und L_2 zwei Listen sind, bezeichnen wir die Verkettung von beiden zu einer neuen Liste L mit $L := L_1, L_2$.

Algorithmus `HNE-reduzibel`

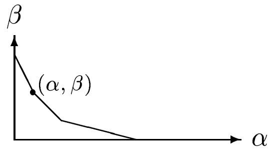
Eingabe: $f \in k[[x,y]]$ quadratfrei und keine Einheit; dabei sei k algebraisch abgeschlossen.

Ausgabe: Zwei Auflösungsbäume in Form von verschachtelten Listen: L_1 ist ein Auflösungsbaum von $f(x,y)$ und L_2 ein Auflösungsbaum von $f/g\,(y,x)$, wobei g der Faktor von f ist, der zur Seite von $N(f)$ mit Steigung -1 gehört.[5]

Schritt 1: `falls` $y \mid f$ `dann` $f \leftarrow f/y, \quad L_1 \leftarrow$ `("Ende")`
`falls` $x \mid f$ `dann` $f \leftarrow f/x, \quad L_2 \leftarrow$ `("Ende")`
// Ab jetzt gilt die Generalvoraussetzung vom Anfang dieses Abschnittes

Schritt 2: Berechne den Eckpunkt (α, β) von $N(f)$, für den alle Seiten von $N(f)$ mit Steigung ≥ -1 unterhalb und alle anderen oberhalb liegen.

Schritt 3: $L_1 \leftarrow L_1, \mathtt{HN}(f, \beta)$
$f \leftarrow f(y,x)$
$L_2 \leftarrow L_2, \mathtt{HN}(f, \alpha)$



Funktion `HN`(f, I)

Parameter: $f \in k[[x,y]]$ quadratfrei, $x \nmid f$; $I \in \mathbb{N}$ so daß alle Seiten mit Abstand[6] $\leq I$ von der Abszisse eine Steigung zwischen -1 und 0 haben.

Rückgabe: Ein Auflösungsbaum aller Faktoren von f, deren Seite in $N(f)$ höchstens den Abstand I von der Abszisse hat.

Start
$L \leftarrow ()$ // leere Liste
`falls` $f(x,0) = 0$ `dann`
$L \leftarrow$ `("Ende")`
$f \leftarrow f/y$
$I \leftarrow I - 1$
`für` jede Seite Δ von $N(f)$ mit Abstand[6] $\leq I$ von der Abszisse :
$L \leftarrow L, \mathtt{HN}\Delta(\Delta, f)$
`return` L
Ende

[5] L_1 enthält also alle Zweige von f mit transversalem $[x]$ und L_2 alle übrigen Zweige.
[6] gemeint ist $\beta \in \mathbb{N} : (\alpha, \beta)$ ist der obere Eckpunkt der Seite

Funktion $\mathtt{HN\Delta}(\Delta, f)$

Parameter: $f \in k[[x,y]]$ quadratfrei, Δ eine Seite aus $N(f)$ mit Steigung ≥ -1

Rückgabe: Ein Auflösungsbaum des zu Δ gehörenden Faktors von f

Start
$M \leftarrow$ Länge(Δ)
$N \leftarrow$ Höhe(Δ)
`falls` $N = 1$ `dann`
  $\delta \leftarrow$ die (einzige) Nullstelle von ϕ_Δ
  `return` $\big(M, 1, (\delta, f)\big)$
`sonst`
  berechne M', N', σ, τ gemäß Lemma 3.11
  $L \leftarrow ()$
  `für` jede Nullstelle δ_i von ϕ_Δ :
    $e_i \leftarrow$ Vielfachheit der Nullstelle δ_i
    `falls` $N\sigma > M\tau$ `dann` $\delta'_i \leftarrow \delta_i$ `sonst` $\delta'_i \leftarrow 1/\delta_i$
    $L \leftarrow L, \big(\delta'_i, \mathtt{HN}(T_2(f, \delta'_i, M', N'), e_i)\,\big)$
  `return` (M', N', L)
Ende

Die folgenden Sätze und Lemmata dienen der Verifikation dieses Algorithmus.

Lemma 3.20 Wir betrachten zwei im Verlauf des Algorithmus aufeinanderfolgende Transformationen:

$$f_1 = T_2\left(f_0, \delta_i^0, M_0', N_0'\right), \qquad f_2 = T_2\left(f_1, \delta_j^1, M_1', N_1'\right)$$

Es seien e_i^0, e_j^1 die Vielfachheiten der Nullstellen δ_i^0, δ_j^1 der entsprechenden charakteristischen Polynome. Dann gilt:

$$\text{Falls} \quad f_2 = T_1\left(f_1, \delta_j^1, M_1'\right), \quad \text{dann ist } e_j^1 \leq e_i^0, \text{ sonst ist sogar } e_j^1 < e_i^0.$$

Beweis: Im Newtonpolygon von f_1 wird nur der Teil bis zur Höhe e_i^0 berücksichtigt, daher hat die nächste ausgewählte Seite Δ_1, mit der $\mathtt{HN\Delta}$ aufgerufen wird, eine Höhe $N_1 \leq e_i^0$. Darum ist

$$e_i^0 \geq N_1 \geq \mathrm{ggT}(M_1, N_1) \geq e_j^1 \tag{3.18}$$

Falls $e_i^0 = e_j^1$ ist, so folgt daraus $N_1 = \mathrm{ggT}(M_1, N_1) \Rightarrow f_1 = T_1\left(f_1, \delta_j^1, M_1'\right)$ ▧

Satz 3.21 *Wir wählen aus der Rückgabe von* $\mathtt{HN}(f, \beta)$ *in Schritt 3 von* **`HNE-reduzibel`** *eine charakteristische Liste aus (vgl. Def. 13) und verlängern*[7] *sie zu einer Hamburger-Noether-Liste wie in der Bemerkung auf S. 82 geschildert. Dann gilt: Es gibt einen irreduziblen Faktor* f_i *von* f*, so daß diese Liste die Hamburger-Noether-Liste von* f_i *ist (bezgl.* $[x]$ *als transversalem Parameter).*

[7] da wir noch nicht gezeigt haben, daß `HNE-reduzibel` terminiert, könnte die Liste auch unendlich lang sein; dann wäre nichts zu tun

Beweis: Sei

$$(*)\qquad \begin{array}{l} (M_1', N_1', (\delta_1' \downarrow))\\ \quad (M_2', N_2', (\delta_2' \downarrow))\\ \qquad\quad \cdots\cdots \\ \qquad\quad (M_i', N_i', (\delta_i' \downarrow))\\ \qquad\qquad\quad \cdots\cdots \end{array}$$

die so konstruierte Hamburger-Noether-Liste (evtl. auch endlich mit letztem Element (`"Ende"`)),

sei $(x(t), y(t))$ die daraus (über die zugehörige HNE) gewonnene Parametrisierung. Wir haben zu zeigen: $f(x(t), y(t)) = 0$.

(i) Nach Lemma 3.15 genügt es, T und T_1 zu untersuchen (aus denen T_2 zusammengesetzt ist):

Sei $g \in k[\![x, y]\!]$ beliebig. Falls $(x(t), y(t))$ eine Parametrisierung von $T(g, M, N)$ ist, dann folgt

$$0 = T(g, M, N)(x(t), y(t)) = \frac{g\left(y(t), x(t)y(t)^Q\right)}{y(t)^s}$$

für ein geeignetes $s \in \mathbb{N}$ und $M = QN + R, \quad 0 < R < M$.

$\Longrightarrow \quad g\left(y(t), x(t)y(t)^Q\right) = 0$

$\Longrightarrow \quad \left(y(t), x(t)y(t)^Q\right)$ ist eine Parametrisierung von g.

Falls $(x(t), y(t))$ eine Parametrisierung von $T_1(g, \delta, Q)$ ist, dann erhalten wir aus

$$0 = T_1(g, \delta, Q)(x(t), y(t)) = g\left(x(t), y(t) + \delta x(t)^Q\right)$$

daß $\left(x(t), y(t) + \delta x(t)^Q\right)$ eine Parametrisierung von g ist.

(ii) Für $j \geq 1$ betrachten wir die Teilliste

$$\begin{array}{l} (M_j', N_j', (\delta_j' \downarrow))\\ \quad (M_{j+1}', N_{j+1}', (\delta_{j+1}' \downarrow))\\ \qquad\qquad \cdots\cdots \end{array}$$

von $(*)$.

Auch diese kann man in eine HNE umrechnen und daraus eine Parametrisierung erhalten, die wir mit $(x_j(t), y_j(t))$ bezeichnen wollen.

Aus (i) folgt: Ist $g \in k[\![x, y]\!]$, so daß $(x_{j+1}(t), y_{j+1}(t))$ eine Parametrisierung von $T_2(g, \delta_j', M_j', N_j')$ ist, dann ist $(x_j(t), y_j(t))$ eine Parametrisierung von g.

(iii) Wir definieren

$$f_1 := f, \qquad f_{i+1} := T_2(f_i, \delta_i', M_i', N_i') \quad \forall\, i \geq 1$$

Fall 1: Die HNL $(*)$ endet mit

$$\begin{array}{l} \qquad \cdots\cdots \\ (M_k', N_k', (\delta_k' \downarrow))\\ \qquad\quad (\texttt{"Ende"}) \end{array}$$

Dies setzt voraus, daß gilt:

$$y \mid f_{k+1} = T_2(f_k, \delta_k', M_k', N_k') \quad \text{und} \quad (x_{k+1}(t), y_{k+1}(t)) = (t, 0)$$

(lt. Def. von (x_{k+1}, y_{k+1})). Offensichtlich ist das eine Parametrisierung von f_{k+1}, also folgt mit Induktion aus (ii), daß $(x(t), y(t)) = (x_1(t), y_1(t))$ eine Parametrisierung von $f_1 = f$ ist.

Fall 2: Die HNL $(*)$ bricht nicht ab. Nach Lemma 3.20 fällt die Folge (e_i) der Vielfachheiten von δ'_i monoton, d. h. es gibt ein $i_0 \geq 1 : e_i = e_{i_0}\ \forall\, i \geq i_0$. Dies impliziert nach dem Lemma insbesondere

$$T_2\left(\cdot, \delta'_i, M'_i, N'_i\right) = T_1\left(\cdot, \delta'_i, M'_i\right) \quad \forall\, i \geq i_0$$

Man beachte ferner, daß aus der Konstanz der e_i folgt, daß unterhalb des Knotens $(M'_{i_0}, N'_{i_0}, (\delta'_{i_0}\downarrow))$ keine Verzweigungen mehr im Baum auftreten, insbesondere besteht der zu berücksichtigende Teil von $N(f_i)$ nur aus einer Seite der (konstanten) Höhe $N = e$.

Wir definieren $$x'(t) := t, \quad y'(t) := \sum_{i \geq i_0} \delta'_i t^{M'_i}$$

$y'(t)$ existiert (d. h. die Reihe konvergiert in der Krull-Topologie) wegen Eigenschaft 3 von T_1 (vgl. Abb. 3.2): Der obere Endpunkt $B =: (a, b)$ der untersten Seite Δ von $N(f_i)$ bleibt fest, so wie auch alle übrigen Seiten. Wegen $e_i = e_{i+1}$ fällt der Punkt A im Bild 3.2 mit B zusammen; Δ wird in $N(f_{i+1})$ ersetzt durch eine Seite mit echt größerer Steigung als Δ.

Also ist $$M_{i+1} > M_i \quad \forall\, i \geq i_0.$$

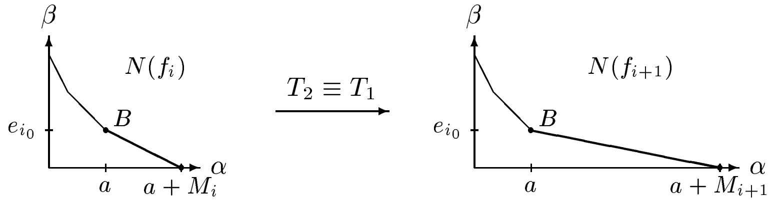
Nach Voraussetzung gilt für beliebiges $y \in k[t]$:

$$\forall\, i \geq i_0 :\ f_{i+1}(t, y) = T_1\left(f_i, \delta'_i, M'_i\right)(t, y) = f_i\left(t, y + \delta'_i\, t^{M'_i}\right)$$

Mit vollständiger Induktion folgt

$$\forall\, i \geq i_0 :\ f_{i+1}(t, 0) = f_{i_0}\left(t\, , \sum_{j=i_0}^{i} \delta'_j\, t^{M'_j}\right)$$

$$\Longrightarrow \quad \forall\, i \geq i_0 :\ f_{i_0}\left(t\, , \sum_{j=i_0}^{i} \delta'_j\, t^{M'_j}\right) \in \langle\, t\, \rangle^{a+M_{i+1}}$$



$$\Longrightarrow \quad f_{i_0}\left(x'(t), y'(t)\right) = f_{i_0}\left(t\, , \sum_{j \geq i_0} \delta'_j\, t^{M'_j}\right) = 0$$

Die aus der Teilliste $(M'_{i_0}, N'_{i_0}, (\delta'_{i_0}\downarrow))$ von $(*)$ gewonnene HNE

$(M'_{i_0+1}, N'_{i_0+1}, (\delta'_{i_0+1}\downarrow))$

$\cdots\cdots$

lautet $$z_{-1} = \delta'_{i_0} z_0^{M'_{i_0}} + \delta'_{i_0+1} z_0^{M'_{i_0+1}} + \ldots$$

darum ist $(x_{i_0}(t), y_{i_0}(t)) = (x'(t), y'(t))$, also eine Parametrisierung von f_{i_0}. Mit Induktion folgt daraus wie in Fall 1, daß $(x(t), y(t))$ eine Parametrisierung von f ist. ▧

Satz 3.22 *Der Algorithmus* `HNE-reduzibel` *terminiert.*

Beweis: Weil alle e_i nach unten durch 1 beschränkt sind, unterscheiden sich beim Ausführen des Algorithmus wegen Lemma 3.20 nur endlich viele Transformationen T_2 von T_1.

Angenommen `HNE-reduzibel` terminiert nicht, dann gibt es Folgen

$$f_i \in k[\![x, y]\!], \quad M_i \in \mathbb{N} \quad \text{und} \quad \delta_i \in k^*,$$

so daß für alle i gilt: $f_{i+1} = T_1(f_i, \delta_i, M_i)$ und die Vielfachheit von δ_i ist eine Konstante $e > 1$. Aus (3.18) folgt, daß bei allen Newtonpolygonen $N(f_i)$ nur je eine Seite berücksichtigt wird, und diese hat immer die Höhe $N = e$.

Aufgrund von Satz 3.21 ist $\left(t\,, \sum_{i\geq 1} \delta_i\, t^{M_i}\right)$ die Parametrisierung eines irreduziblen Faktors g_1 von f_1. Dessen Newtonpolygon hat die Höhe $1 < e = N$. Wir definieren $g_{i+1} := T_1(g_i, \delta_i, M_i)$ und stellen fest, daß für alle i gilt: Die unterste Seite von $N(f_i/g_i)$ hat die gleiche Steigung wie die von $N(f_i)$ und die Höhe $e-1$, das charakteristische Polynom zu dieser Seite lautet $c(\lambda - \delta_i)^{e-1}$ (bei $N(f_i)$ war es nämlich $c'(\lambda - \delta_i)^e$) und $T_1(f_i/g_i, \delta_i, M_i) = f_{i+1}/g_{i+1}$.

Wieder nach Satz 3.21 ist $\left(t\,, \sum_{i\geq 1} \delta_i\, t^{M_i}\right)$ auch eine Parametrisierung von f_1/g_1 $\Rightarrow$ $g_1 \mid f_1/g_1$ $\Rightarrow$ $g_1^2 \mid f_1$. Wenn aber f_1 nicht quadratfrei ist, dann ist auch die ursprünglich gegebene Potenzreihe nicht quadratfrei gewesen (Lemmata 3.17, 3.6, 3.12), im Widerspruch zur Voraussetzung des Algorithmus. ▧

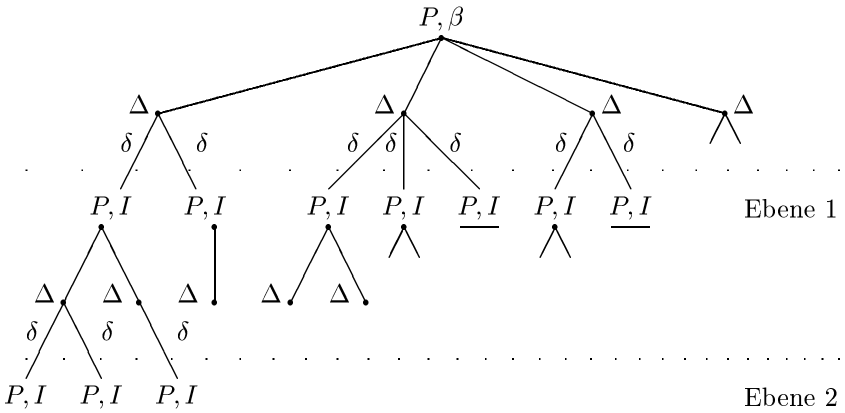
Lemma 3.23 *Sei r die Anzahl der Blätter eines von* `HNE-reduzibel` *berechneten Baumes B, dann sind die r charakteristischen Listen, die man beim Zurückverfolgen von den Blättern zur Wurzel von B erhält (vgl. Def. 13), paarweise verschieden.*

Beweis: Nach Konstruktion in `HNΔ` sind alle Tripel (M_i', N_i', δ_j') innerhalb des selben Knotens von B verschieden. ▧

Satz 3.24 *Die Funktion* `HN`(f, β) *– aufgerufen in Schritt 3 von* `HNE-reduzibel` *– gibt einen Auflösungsbaum zurück, der genauso viele Blätter besitzt wie f irreduzible Faktoren mit transversalem $[x]$.*

Beweis: Falls f gar keinen solchen Faktor besitzt, ist $\beta = 0$ und das Ergebnis von `HN`(f, β) ein leerer Baum, also 0 Blätter. Sei darum o.B.d.A. $\beta \geq 1$.

Dem (nicht leeren) Auflösungsbaum, den `HN`(f, β) zurückgibt, entspricht ein Baum



Hierbei steht P für Polygone (die zu betrachtenden Teile der Newtonpolygone), $I = I(P)$ ist die Höhe von P, Δ sind die Seiten des darüberliegenden Polygons P und δ bezeichnet die Nullstellen des charakteristischen Polynoms ϕ_Δ der im Baum darüberstehenden Seite. Ein unterstrichener Eintrag P, I markiert ein Blatt des Baumes. Man beachte, daß per definitionem von `HNΔ` gilt: $I(P)$ ist die Vielfachheit e_i der darüber befindlichen Nullstelle δ von ϕ_Δ.

Ferner bezeichne $M(\Delta)$ die Länge und $N(\Delta)$ die Höhe der Seite Δ,

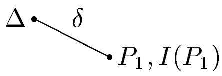
$$e(\Delta) := \mathrm{ggT}(M(\Delta), N(\Delta)), \quad M'(\Delta) := \frac{M(\Delta)}{e(\Delta)}, \quad N'(\Delta) := \frac{N(\Delta)}{e(\Delta)}$$

Ist P in einem Blatt, dann ist $I(P) = 1$. Für die anderen P aus dem Baum zeigen wir folgende Gleichung:

$$I(P) = \sum_{\substack{\Delta \text{ im Baum} \\ \text{unterhalb von } P}} N(\Delta) = \sum_{\Delta \text{ unter } P} \sum_{\substack{P_1 \\ P_1 \text{ unter } \Delta}} N'(\Delta)\, I(P_1) \tag{3.19}$$

Trivialerweise ist $\quad I(P) = \sum_{\Delta \in P} N(\Delta) = \sum_{\Delta \text{ unter } P} N(\Delta)\,.$

Und wie schon bemerkt ist $I(P_1)$ gleich der Vielfachheit der Nullstelle δ von ϕ_Δ, δ im Baum zwischen Δ und P_1.



Daraus folgt:

$$\sum_{P_1 \text{ unter } \Delta} I(P_1) = \deg \phi_\Delta = e(\Delta)$$

(vergleiche Lemmata 3.14 & 3.19), und hieraus wiederum

$$\sum_{P_1 \text{ unter } \Delta} N'(\Delta)\, I(P_1) = N'(\Delta)\, e(\Delta) = N(\Delta)$$

Als nächstes wählen wir ein Blatt s aus dem Baum aus. Nach Satz 3.21 gibt es einen irreduziblen Faktor f_s von f, so daß die Liste, die man beim Durchlaufen des Baumes bis zu diesem Blatt erhält, eine charakteristische Liste von f_s ist. Aus

Lemma 3.23 folgt $s \neq t \;\Rightarrow\; f_s \neq f_t$. Daher ist die Behauptung unseres Satzes ein Resultat folgender Gleichung, die wir noch zu zeigen haben:

$$\sum_{\substack{g \text{ irred. Faktor von } f \\ \text{so daß } [x] \text{ transversal}}} e\big(k[\![x,y]\!] \,/\, \langle g\rangle\big) \;=\; \sum_{s \text{ Blatt des Baumes}} e\big(k[\![x,y]\!] \,/\, \langle f_s\rangle\big) \tag{3.20}$$

Zur Erinnerung: $e\big(k[\![x,y]\!] \,/\, \langle g\rangle\big)$ ist die Multiplizität der Kurve $k[\![x,y]\!] \,/\, \langle g\rangle$; wenn wie hier vorausgesetzt g irreduzibel ist und $[x] \in k[\![x,y]\!] \,/\, \langle g\rangle$ transversal, dann ist die Multiplizität gleich der Höhe des Newtonpolygons von g. Daher ist die linke Seite von (3.20) gleich β. Wir zeigen mit vollständiger Induktion über die Anzahl j der Ebenen des Baumes, daß auch die rechte Seite von (3.20) gleich β ist.

I.A.: $j = 0$: Dann besteht der Baum nur aus der Wurzel ($=1$ Blatt s) und β ist 1. Das Newtonpolygon des zum Blatt gehörenden Faktors f_s hat die Höhe $1 = \beta \quad \Rightarrow \; e\big(k[\![x,y]\!] \,/\, \langle f_s\rangle\big) = 1 = \beta = I(P)$.

I.S.: $j \geq 1$: Sei P_0 die Wurzel des Baumes, P_1 ein Polygon aus Ebene 1, t ein Blatt aus dem Teilbaum mit Wurzel P_1. Die charakteristische Liste, die man durch Durchlaufen des Teilbaumes von P_1 bis t erhält, ist eine charakteristische Liste von $T_2(f_t, \delta, M'(\Delta), N'(\Delta)) =: g_t \quad (\Delta, \delta$ im Baum zwischen P_0 und $P_1)$.

Das Newtonpolygon von f_t ist eine Seite mit gleicher Steigung wie Δ. Seien M_t, N_t die Länge bzw. Höhe dieser Seite, $e_t := \mathrm{ggT}(M_t, N_t)$, dann gilt folglich

$$N_t/e_t = N(\Delta)/e(\Delta) = N'(\Delta).$$

Nach Lemmata 3.6 & 3.12 ist e_t die Höhe des Newtonpolygons von g_t; weil für beide Potenzreihen $[x]$ transversal ist, ist die Multiplizität der durch f_t bzw. g_t definierten Kurve gleich der Höhe des entsprechenden Newtonpolygons. Es folgt

$$e\big(k[\![x,y]\!] \,/\, \langle f_t\rangle\big) = N_t = N'(\Delta)\, e_t = N'(\Delta)\, e\big(k[\![x,y]\!] \,/\, \langle g_t\rangle\big) \tag{3.21}$$

Die Induktionsvoraussetzung lautet:

$$I(P_1) \;=\; \sum_{t \text{ Blatt des Teilbaumes}} e\big(k[\![x,y]\!] \,/\, \langle g_t\rangle\big)$$

Aus den Gleichungen (3.19) und (3.21) erhalten wir

$$\begin{aligned}
\beta = I(P_0) \;&=\; \sum_{\Delta \text{ unter } P_0} \; \sum_{P_1 \text{ unter } \Delta} N'(\Delta) \sum_{t \text{ Blatt unter } P_1} e\big(k[\![x,y]\!] \,/\, \langle g_t\rangle\big) \\
&=\; \sum_{\Delta \text{ unter } P_0} \; \sum_{t \text{ Blatt unter } \Delta} N'(\Delta)\, e\big(k[\![x,y]\!] \,/\, \langle g_t\rangle\big) \\
&=\; \sum_{\Delta \text{ unter } P_0} \; \sum_{t \text{ Blatt unter } \Delta} e\big(k[\![x,y]\!] \,/\, \langle f_t\rangle\big) \\
&=\; \sum_{t \text{ Blatt des Baumes}} e\big(k[\![x,y]\!] \,/\, \langle f_t\rangle\big)
\end{aligned}$$

Damit ist (3.20) bewiesen und wir sind fertig. ▧

Nun können wir die Korrektheit von `HNE-reduzibel` beweisen. Der Algorithmus terminiert nach Satz 3.22, und es gilt:

Satz 3.25 *Die vom Algorithmus* `HNE-reduzibel` *zurückgegebenen Bäume L_1 und L_2 haben folgende Eigenschaften:*

- *Für jeden irreduziblen Faktor f_i von f, für den $[x]_{f_i}$ transversal ist, gibt es genau ein Blatt b von L_1, so daß die beim Durchlaufen des Baumes nach b konstruierte Liste eine charakteristische Liste von f_i ist. Für jedes Blatt b von L_1 gibt es genau einen irreduziblen Faktor f_i von f mit transversalem $[x]_{f_i}$, so daß die aus b erhaltene Liste eine charakteristische Liste von f_i ist.*
- *Der gleiche Zusammenhang besteht zwischen den Blättern von L_2 und den irreduziblen Faktoren von f, für die $[x]$* nicht *transversal ist.*

Mit anderen Worten: L_1 und L_2 haben die Eigenschaften, die für die Ausgabe von `HNE-reduzibel` *gefordert werden.*

Beweis: Wenn L_1 Auflösungsbaum von f ist, dann ist trivialerweise L_1, (`"Ende"`) ein Auflösungsbaum von $y \cdot f$. Analoges gilt für L_2 im Zusammenhang mit $x \cdot f$. Damit ist der Sonderfall, daß $N(f)$ Halbgeraden enthält, abgedeckt (siehe Schritt 1).

Für jedes Blatt von L_1 gibt es einen Faktor mit den gewünschten Eigenschaften wegen Satz 3.21. Für jeden Faktor gibt es höchstens ein passendes Blatt nach Satz 2.2 und Lemma 3.23. Dank Satz 3.24 folgt daraus, daß es für jeden Faktor auch mindestens ein Blatt und für jedes Blatt höchstens einen Faktor gibt.

Nach Wahl von (α, β) in Schritt 2 des Algorithmus ist (β, α) derjenige Punkt im Newtondiagramm von $f(y, x)$, für den gilt: Alle Seiten von $N(f(y, x))$ mit Steigung > -1 liegen unterhalb von (β, α), alle anderen oberhalb. Wir bezeichnen mit g den Faktor von f, dessen Newtonpolygon die Seite von $N(f)$ mit Steigung -1 ist ($g := 1$ falls $N(f)$ keine Seite dieser Steigung hat). Dann hat $N(f/g)$ keine Seite der Steigung -1 mehr, d. h. L_2 ist identisch mit der Ausgabe L_1 von `HNE-reduzibel` angewendet auf $(f/g)(y, x)$. Unter Verwendung des bereits Gezeigten sehen wir, daß die gewünschte 1:1-Beziehung zwischen den Blättern von L_2 und den irreduziblen Faktoren f_i von f/g existiert, für die $[y]_{f_i}$ transversal ist. Nach Lemma 3.1 sind das genau die Faktoren von f, für die $[x]_{f_i}$ nicht transversal ist. ▧

3.2.4 Beispiele

Es folgen zum Schluß dieses Abschnittes noch ein paar Beispiele für den Algorithmus:

Beispiel 1: Wir betrachten zum dritten Mal die (irreduzible) Kurve

$$f = y^2 + x^7 y + x^8 + x^9, \quad k = \overline{\mathbb{F}_2}$$

(vgl. Beispiel 1 auf S. 60 und S. 69). Das Ergebnis ist

$$L_1 = \mathtt{HN}(f, 2), \quad L_2 = \mathtt{HN}(f(y, x), 0) = ()$$

Die einzelnen Transformationen von f zur Berechnung von $\mathtt{HN}(f,2)$ sind auf Seite 69 nachzulesen; der irreduzible Fall zeichnet sich dadurch aus, daß zu jeder Zeit $N(f_i)$ nur aus einer Seite Δ_i besteht und ϕ_{Δ_i} nur eine (evtl. mehrfache) Nullstelle hat, d. h. der Baum unverzweigt ist.

$$\begin{array}{rl}\mathtt{HN}(f,2) = & \big((4,1,(1\downarrow))\big) \\ & \quad\big((9,2,(1\downarrow))\big) \\ & \qquad\Big((6,1,(1,\ y+(1+x)\left(y+x^2\right)^3\Big)\end{array}$$

Beispiel 2: Als nächstes nehmen wir das Beispiel, für das `HNE-irreduzibel` nur „f reduzibel!“ als Ergebnis geliefert hatte. Man beachte, daß `HNE-reduzibel` einen algebraisch abgeschlossenen Körper voraussetzt. Wir betrachten also

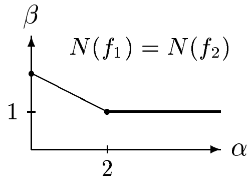
$$f := y^4 - x^6 \in \mathbb{C}[[x,y]]$$

Wieder genügt ein Auflösungsbaum, um f vollständig zu entwickeln:
$(\alpha,\beta) = (0,4)$, $\quad L_1 = \mathtt{HN}(f,4)$, $\quad L_2 = ()$.

Bearbeitung von $\mathtt{HN}(f,4)$: $\quad M = 6, \quad N = 4 > 1, \quad M' = 3, \quad N' = 2$
$\phi_\Delta(\lambda) = \lambda^2 - 1 = (\lambda-1)(\lambda+1) \rightsquigarrow e_1 = e_2 = 1, \quad \delta_1 = 1,\ \delta_2 = -1$
$3 = 1\cdot 2 + 1,\ 2 = 2\cdot 1 \rightsquigarrow \delta_1' = 1/1 = 1,\ \delta_2' = 1/{-1} = -1$

$$\begin{aligned} f_1 &:= T_2(f,1,3,2) = T_1\left(\frac{x^4y^4 - y^6}{y^4},1,2\right) = x^4 - (y+x^2)^2 = -y^2 - 2x^2y \\ f_2 &:= T_2(f,-1,3,2) = T_1(x^4 - y^2,-1,2) = x^4 - (y-x^2)^2 = -y^2 + 2x^2y \end{aligned}$$

Bearbeitung von $\mathtt{HN}(f_1,1)$ und $\mathtt{HN}(f_2,1)$: Von den Newtonpolygonen der beiden Transformierten wird nur die Halbgerade betrachtet:

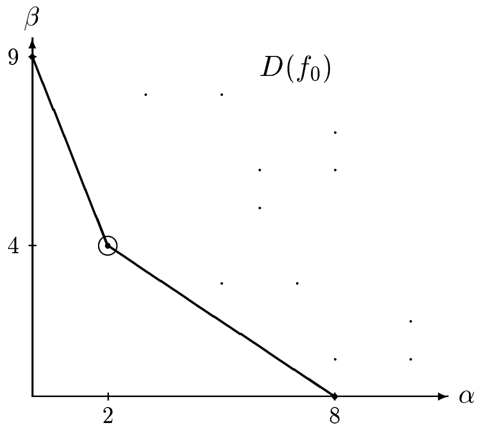


$$f_{1,2}(x,0) = 0 \rightsquigarrow L_1 = \big((3,2,(1\swarrow),(-1\searrow))\big) \qquad \big((\texttt{"Ende"})\big)\ \big((\texttt{"Ende"})\big)$$

Daraus lassen sich die beiden HNEs von f

$$\begin{array}{rcl}[y] = z_{-1} & = & z_0 z_1 \\ [x] = z_0 & = & z_1^2\end{array} \quad \text{und} \quad \begin{array}{rcl}[y] = z_{-1} & = & z_0 z_1 \\ [x] = z_0 & = & -z_1^2\end{array} \quad \text{ablesen.}$$

Beispiel 3: $\quad f := x(y^2+x^3y+x^3)(y^2+x^5y+x^3)(y^5+x^2) \in \overline{\mathbb{F}_2}[[x,y]]$
Schritt 1: $f_0 := (y^2+x^3y+x^3)(y^2+x^5y+x^3)(y^5+x^2)$, $\quad L_2 :=$ (`"Ende"`)



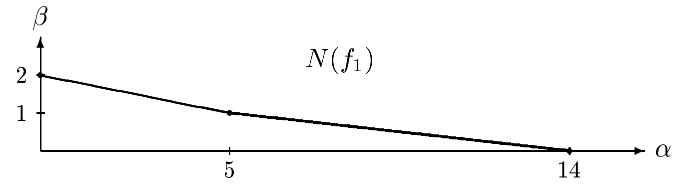
Schritt 2: $\quad (\alpha, \beta) = (2, 4)$

Schritt 3: Bearbeitung von $\mathtt{HN}(f_0, 4)$:

$$\begin{aligned} &M = 6, && N = 4 > 1 \\ &M' = 3, && N' = 2 \\ &L_\Delta(f_0) = x^2(y^4 + x^6) \\ &\phi_\Delta(\lambda) = \lambda^2 + 1 = (\lambda + 1)^2 \\ &\rightsquigarrow\ e_1 = 2, \quad \delta_1 = 1 \end{aligned}$$

$$\begin{aligned} f_1 \ &:= \ T_2(f_0, 1, 3, 2) = T_1\left(\frac{(x^2y^2 + xy^4 + y^3)(x^2y^2 + xy^6 + y^3)(x^5y^5 + y^2)}{y^6}, 1, 2\right) \\ &= \ (y + xy^2 + x^5)(y + xy^4 + x^9)(1 + x^5y^3 + x^7y^2 + x^9y + x^{11}) \end{aligned}$$

Bearbeitung von $\mathtt{HN}(f_1, 2)$:



$M_1 = 5, \quad N_1 = 1, \quad M_2 = 9, \quad N_2 = 1$
$L_{\Delta_1}(f_1) = y(y + x^5), \quad L_{\Delta_2}(f_1) = x^5(y + x^9) \quad \rightsquigarrow \quad \delta_1 = 1 = \delta_2$

Bearbeitung von $\mathtt{HN}(f_0(y, x), 2):\quad M = 5, \quad N = 2 > 1$
$L_{\Delta'}(f_0(y, x)) = x^4(y^2 + x^5) \ \rightsquigarrow\ e_1 = 1, \quad \delta_1 = 1$

$$\begin{aligned} f_2 \ &:= \ T_2(f_0(y, x), 1, 5, 2) \\ &= \ T_1\left(\frac{(y^2 + x^3y^7 + x^3y^6)(y^2 + x^5y^{11} + x^3y^6)(y^5 + x^2y^4)}{y^8}, 1, 2\right) \\ &= \ \left(1 + x^3(y + x^2)^5 + x^3(y + x^2)^4\right)\left(1 + x^5(y + x^2)^9 + x^3(y + x^2)^4\right) y \end{aligned}$$

Bearbeitung von $\mathtt{HN}(f_2, 1):\quad f_2(x, 0) = 0$; wir sind fertig.

Ergebnis: Die beiden Auflösungsbäume von f (und $(f/g)(y, x)$ genaugenommen) lauten

$$L_1 = \big((3, 2, (1 \downarrow))\big) \qquad \text{und} \qquad L_2 = \big((\texttt{"Ende"}),\ (5, 2, (1 \downarrow))\big)$$
$$\big((5, 1, (1, f_1)),\ (9, 1, (1, f_1))\big) \qquad\qquad \big((\texttt{"Ende"})\big)$$

Dazu gehören die zweimal zwei Hamburger-Noether-Entwicklungen

$$\begin{aligned} z_{-1} &= z_0 z_1 \\ z_0 &= z_1^2 + z_1^5 + \ldots \end{aligned} \qquad \text{und} \qquad \begin{aligned} z_{-1} &= z_0 z_1 \\ z_0 &= z_1^2 + z_1^9 + \ldots \end{aligned}$$

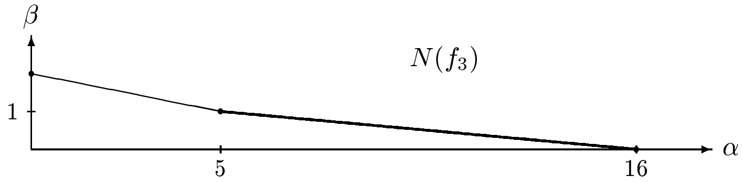
mit $z_{-1} = [y]$, $z_0 = [x]$, und

$$z_{-1} = 0 \quad \text{und} \quad \begin{array}{rcl} z_{-1} & = & z_0^2 z_1 \\ z_0 & = & z_1^2 \end{array}$$

mit $z_{-1} = [x]$, $z_0 = [y]$.

Bemerkung: Interessiert man sich jetzt z. B. speziell für den ersten Zweig und möchte dessen HNE fortsetzen, kann man die Information aus dem korrespondierenden Blatt $(5, 1, (1, f_1))$ verwenden, um die Berechnung weiterzuführen. Die nächste auszuführende Transformation ist

$$T_2(f_1, 1, 5, 1) = T_1(f_1, 1, 5) = (y + xy^2 + x^{11})(y + x^5 + xy^4 + x^9 + x^{21})(1 + \ldots) =: f_3$$



Nur die unterste Seite wird berücksichtigt (vgl. Bemerkung auf Seite 82), d. h. $M = 11, \quad N = 1, \quad L_\Delta(f_3) = x^5(y + x^{11}) \quad \rightsquigarrow \quad \delta = 1.$

Die um einen Knoten erweiterte charakteristische Liste des ausgewählten irreduziblen Faktors ist folglich

$$\begin{array}{l} (3, 2, (1 \downarrow)) \\ \quad (5, 1, (1 \downarrow)) \\ \qquad (11, 1, (1, f_3)) \end{array} \quad , \quad \text{die HNE dazu} \quad \begin{array}{rcl} z_{-1} & = & z_0 z_1 \\ z_0 & = & z_1^2 + z_1^5 + z_1^{11} + \ldots \end{array}$$

3.3 Hamburger-Noether-Entwicklung über nicht algebraisch abgeschlossenen Körpern (Berechnung des Primitiven Elements)

Normalerweise hat man bei der Implementation nicht die Möglichkeit, sämtliche Berechnungen über algebraisch abgeschlossenen Körpern auszuführen. Vielmehr wird man mit einem nicht abgeschlossenen Körper wie $\mathbb{Q}$ oder $\mathbb{F}_p$ starten. Glücklicherweise stellt sich heraus, daß wir die HNE über dem algebraischen Abschluß berechnen können, ohne in diesen Körper wechseln zu müssen:

Satz 3.26 *Sei k ein beliebiger Körper, $0 \neq f \in k[[x,y]]$ quadratfrei und keine Einheit. Ist k vollkommen, dann ist $\mathcal{R} := \overline{k}[[x,y]]/\langle f\rangle$ der lokale Ring einer algebroiden Kurve.*

Ist $\mathcal{R}$ der lokale Ring einer algebroiden Kurve (k nicht notwendigerweise vollkommen), dann gibt es eine endliche algebraische *Körpererweiterung $\overline{k} \supseteq \widetilde{k} \supseteq k$, so daß f über $\widetilde{k}[[x,y]]$ in Faktoren f_i zerfällt, die über $\overline{k}[[x,y]]$ irreduzibel sind, und so daß die HNE von $\mathcal{R}_i := \overline{k}[[x,y]]/\langle f_i\rangle$ in der Basis $\{[x]_{f_i},[y]_{f_i}\}$ bereits über $\widetilde{k}$ existiert.*

Wir können uns also darauf beschränken, die HNE über $\widetilde{k}$ zu berechnen. Ziel dieses Abschnittes ist es, den Algorithmus `HNE-reduzibel` so zu modifizieren, daß er ohne die Verwendung von $\overline{k}$ auskommt und stattdessen $\widetilde{k}$ berechnet. Falls k ein endlicher Körper ist oder die Charakteristik Null hat, ist die Darstellung von $\widetilde{k}$ mit Hilfe eines primitiven Elements möglich. Wir werden ein Verfahren entwickeln, wie man dieses unter Verwendung von Standardbasen erhalten kann.

Beweis von 3.26: $\mathcal{R}$ ist der lokale Ring einer algebroiden Kurve, denn f ist auch quadratfrei über $\overline{k}[[x,y]]$ wegen des folgenden Lemmas 3.27 ($\mathrm{ggT}(f, \partial_x f, \partial_y f)$ existiert in $k[[x,y]]$ und ist in jeder Körpererweiterung von k gleich). Die übrigen Eigenschaften, die $\operatorname{Spec}\mathcal{R}$ als ebene algebroide Kurve erfüllen muß, sind leicht einzusehen.

Sei $\operatorname{Spec}\mathcal{R}$ algebroide Kurve und $\mathcal{P}$ die Menge aller charakteristischen Polynome ϕ_Δ, die im Verlauf des Algorithmus `HNE-reduzibel` (angewandt auf $f \in \overline{k}[[x,y]]$) auftreten. Das ist eine endliche Menge, daher ist

$$\widetilde{k} := \text{Zerfällungskörper aller Elemente von } \mathcal{P}$$

eine endliche algebraische Erweiterung von k (die natürlich auch im algebraischen Abschluß $\overline{k}$ von k enthalten ist). Eine Analyse von `HNE-reduzibel` zeigt, daß sämtliche Berechnungen innerhalb von $\widetilde{k}$ ausgeführt werden können, auch wenn man das Abbruchkriterium $N = 1$ in `HNΔ` wegläßt und die einzelnen HNEs ad infinitum fortsetzt (es treten nur noch lineare charakteristische Polynome auf). Folglich existieren alle Hamburger-Noether-Entwicklungen von $\mathcal{R}$ bereits über $\widetilde{k}$. Wenn n die Anzahl der Zweige von $\mathcal{R}$ ist, erhalten wir also durch Rückwärtseinsetzen n Parametrisierungen von R mit Koeffizienten aus $\widetilde{k}$. Folglich zerfällt f über $\widetilde{k}[[x,y]]$ bereits in (mindestens) n Faktoren, welche irreduzibel über $\overline{k}[[x,y]]$ sein müssen. ▧

Lemma 3.27 *Sei k ein vollkommener Körper, $f \in k[\![x,y]\!]$ oder $f \in k[x,y]$. Es bezeichne $\partial_x f$ die (formale) partielle Ableitung von f nach x, $\partial_y f$ die Ableitung nach y.*

Genau dann ist f quadratfrei (in $k[\![x,y]\!]$ bzw. $k[x,y]$), wenn 1 ein größter gemeinsamer Teiler von f, $\partial_x f$ und $\partial_y f$ ist (in $k[\![x,y]\!]$ bzw. $k[x,y]$).

Außerdem ist $\quad \dfrac{f}{\mathrm{ggT}(f, \partial_x f, \partial_y f)} \quad$ *immer ein quadratfreier Teiler von f.*

Beweis: Es sei $P := k[\![x,y]\!]$ bzw. $P := k[x,y]$ je nach betrachtetem Ring. Falls f nicht quadratfrei (in P) ist, gibt es $g, h \in P$, h keine Einheit, und ein $q > 1$ mit $f = g \cdot h^q$.

$$\Longrightarrow \quad \partial_z f = \partial_z g \cdot h^q + qg \cdot h^{q-1} \cdot \partial_z h \quad \text{für beide } z \in \{x, y\}$$

Daher ist $h \in P$ ein gemeinsamer Teiler von f, $\partial_x f$ und $\partial_y f$, also ist 1 kein größter gemeinsamer Teiler. Falls $f = \prod_i f_i$, quadratfrei ist, f_i irreduzibel, dann ist

$$\partial_z f = \sum_i \partial_z f_i \cdot \prod_{j \neq i} f_j \quad \text{für beide } z \in \{x, y\}$$

Angenommen es gibt einen gemeinsamen Teiler von f, $\partial_x f$ und $\partial_y f$, der keine Einheit ist. Dann gibt es auch ein i, für das f_i ein gemeinsamer Teiler ist. Wegen der Irreduzibilität von f_i und der Quadratfreiheit von f folgt

$$f_i \mid \partial_x f \quad \Longrightarrow \quad f_i \mid \partial_x f_i\,, \qquad f_i \mid \partial_y f \quad \Longrightarrow \quad f_i \mid \partial_y f_i$$

woraus aus Gradgründen wiederum folgt: $\partial_x f_i = 0 = \partial_y f_i$.

Weil k nach Voraussetzung vollkommen ist, ist f_i entweder konstant oder p-te Potenz eines anderen Elements von P, p die Charakteristik von k, im Widerspruch zur Irreduzibilität von f_i. ↯

Also ist jeder gemeinsame Teiler von f, $\partial_x f$ und $\partial_y f$ eine Einheit in P, q.e.d.

Zum Beweis der zweiten Behauptung schreiben wir kurz $g := \mathrm{ggT}(f, \partial_x f, \partial_y f)$. Es folgt:

$$\begin{aligned}
\mathrm{ggT}\left(\frac{f}{g}\,,\ \partial_x \frac{f}{g}\,,\ \partial_y \frac{f}{g}\right) &= \mathrm{ggT}\left(\frac{f}{g}\,,\ \frac{(\partial_x f)g - f\partial_x g}{g^2}\,,\ \frac{(\partial_x f)g - f\partial_x g}{g^2}\right) \\
&= \mathrm{ggT}\left(\frac{f}{g}\,,\ \frac{\partial_x f}{g} - \frac{f}{g}\frac{\partial_x g}{g}\,,\ \frac{\partial_y f}{g} - \frac{f}{g}\frac{\partial_y g}{g}\right) \\
&= \mathrm{ggT}\left(\frac{f}{g}\,,\ \frac{\partial_x f}{g}\,,\ \frac{\partial_y f}{g}\right) \\
&= \frac{\mathrm{ggT}(f, \partial_x f, \partial_y f)}{g} = \frac{g}{g} = 1
\end{aligned}$$

Aus dem ersten Teil folgt darum, daß $\frac{f}{g}$ quadratfrei ist. $\square$

Korollar *$f \in k[x,y]$ quadratfrei* $\Longrightarrow$ *$f \in k[\![x,y]\!]$ quadratfrei*

Beweis: Sei $f \in k[x,y]$ quadratfrei
$\Longrightarrow$ $1 \in \mathrm{ggT}_{k[x,y]}(f, \partial_x f, \partial_y f) \subset \mathrm{ggT}_{k[\![x,y]\!]}(f, \partial_x f, \partial_y f)$
$\Longrightarrow$ $f \in k[\![x,y]\!]$ quadratfrei.

Bemerkung: Am Beweis sieht man bereits, warum die Umkehrung nicht gilt: Falls $f \in k[x,y]$ als Potenzreihe quadratfrei ist, dann folgt lediglich, daß f Produkt aus einem quadratfreien Polynom und einer Einheit im Potenzreihenring ist.

Beispielsweise ist $x(1+x)^2$ quadratfrei als Potenzreihe, aber nicht als Polynom.

Mit Lemma 3.27 und dem zugehörigen Korollar können wir somit feststellen, ob ein gegebenes Polynom $f \in k[x,y]$ eine ebene algebroide Kurve $\operatorname{Spec} \overline{k}[\![x,y]\!]/\langle f \rangle$ definiert, allein mit Hilfe von Rechenoperationen in $k[x,y]$.

Ziel ist es also, eine Körpererweiterung zu finden, in der alle Berechnungen der HNE genauso verlaufen wie im algebraischen Abschluß, in der insbesondere die Kurve in die gleichen Zweige zerfällt. Da die charakteristischen Polynome, die im Laufe der Berechnung auftreten, nicht a priori bekannt sind, starten wir im ursprünglich gegebenen Körper $\widetilde{k} := k$, testen bei jedem charakteristischen Polynom, sobald es berechnet ist, ob es in $\widetilde{k}[\lambda]$ bereits vollständig in Linearfaktoren zerfällt, und ersetzen ggf. $\widetilde{k}$ durch den Zerfällungskörper.

Das bedeutet, daß wir den Algorithmus für algebraisch abgeschlossene Körper bis auf zwei Änderungen vollständig für beliebige Körper übernehmen können. Die eine betrifft die Voraussetzungen bei der Eingabe:

Algorithmus `HNE-reduzibel`

Eingabe: $f \in k[\![x,y]\!]$ quadratfrei und keine Einheit; dabei sei k vollkommen. Oder aber k ist beliebig, und es ist anderweitig gewährleistet, daß $f \in \overline{k}[\![x,y]\!]$ quadratfrei ist (vgl. Satz 3.26).

Ausgabe: Eine endliche algebraische Erweiterung $\widetilde{k}$ von k, über der f vollständig zerfällt, und zwei Auflösungsbäume in Form von verschachtelten Listen mit Koeffizienten in $\widetilde{k}$: L_1 ist ein Auflösungsbaum von $f(x,y)$ (als Element von $\overline{k}[\![x,y]\!]$) und L_2 ein Auflösungsbaum von $f/g\,(y,x)$, wobei g der Faktor von f ist, der zur Seite von $N(f)$ mit Steigung -1 gehört.

Die andere Änderung besteht in einer Ergänzung der

Funktion `HNΔ`(Δ, f)

Parameter: $f \in \widetilde{k}[\![x,y]\!]$ quadratfrei, Δ eine Seite aus $N(f)$ mit Steigung ≥ -1. Dabei ist $\widetilde{k}$ der aktuelle Körper.

Rückgabe: Ein Auflösungsbaum des zu Δ gehörenden Faktors von $f \in \overline{\widetilde{k}}[\![x,y]\!]$

Seiteneffekt: $\widetilde{k}$ wird – falls nötig – durch eine Körpererweiterung ersetzt.

Start

$M \leftarrow$ Länge(Δ)
$N \leftarrow$ Höhe(Δ)
`falls` $N = 1$ `dann`
  $\delta \leftarrow$ die (einzige) Nullstelle von ϕ_Δ
  `return` $\big(M, 1, (\delta, f)\big)$
`sonst`
  berechne M', N', σ, τ gemäß Lemma 3.11
  $L \leftarrow ()$
  `solange` ϕ_Δ über $\widetilde{k}$ nicht vollständig in Linearfaktoren zerfällt $\leftarrow$
    wähle einen nichtlinearen irreduziblen Faktor g von ϕ_Δ $\leftarrow$
    ersetze $\widetilde{k}$ durch $\widetilde{k}[\lambda]/\langle g(\lambda)\rangle$ $\leftarrow$
    und rechne ab sofort im neuen Körper $\widetilde{k}$ weiter
  `für` jede Nullstelle δ_i von ϕ_Δ :
    $e_i \leftarrow$ Vielfachheit der Nullstelle δ_i
    `falls` $N\sigma > M\tau$ `dann` $\delta_i' \leftarrow \delta_i$ `sonst` $\delta_i' \leftarrow 1/\delta_i$
    $L \leftarrow L, \big(\delta_i', \texttt{HN}(T_2(f, \delta_i', M', N'), e_i)\big)$
  `return` (M', N', L)

Ende

3.3.1 Primitive Körpererweiterungen

Wir beschäftigen uns nun näher mit der konkreten Darstellung von $\widetilde{k}$. Da wir den Algorithmus später in Singular implementieren wollen und Singular nur einfache algebraische Erweiterungen kennt, versuchen wir, ein primitives Element für $\widetilde{k}$ zu finden. Zunächst haben wir sicherzustellen, daß ein solches Element überhaupt existiert. Die wichtigsten Körpererweiterungen mit dieser Eigenschaft sind die separablen Körpererweiterungen.

Definition 3.15 Sei k ein beliebiger Körper. Eine Körpererweiterung L von k heißt **primitiv** (über k), wenn es ein $x \in L$ gibt, so daß $L = k(x)$. In diesem Fall heißt x ein **primitives Element**.

Eine algebraische Körpererweiterung $L \supset k$ heißt **separabel**, wenn für jedes Element $\alpha \in L$ das zugehörige Minimalpolynom $f \in k[x]$ nur einfache Nullstellen im algebraischen Abschluß von k hat.

Bemerkung: Das ist äquivalent dazu, daß α keine Nullstelle der Ableitung f' ist ($(x-\alpha)^2 \mid f \Leftrightarrow (x-\alpha) \mid f', f$), was wiederum äquivalent ist mit $f' \neq 0$, denn f ist in $k[x]$ irreduzibel:

$$f' \neq 0 \implies \deg(\mathrm{ggT}(f, f')) \leq \deg(f') < \deg(f) \implies \mathrm{ggT}(f, f') \in k[x]$$

$$\text{ist ein echter Teiler von } f \implies \mathrm{ggT}(f, f')(\alpha) \neq 0 \implies f'(\alpha) \neq 0$$

Separable Körpererweiterungen sind im Grunde schon genug für unsere Zwecke:

Lemma 3.28 *Sei $L \supset k$ eine algebraische Körpererweiterung. Dann gilt:*

(i) $\operatorname{char} k = 0 \implies$ *L ist separabel über k*
(ii) *k ist endlich* $\implies$ *L ist separabel über k*

Beweis:

(i) folgt direkt aus der Bemerkung, denn in Charakteristik 0 verschwindet die Ableitung eines nichtkonstanten Polynoms nicht.

(ii) Seien $f \in k[x]$ mit $f' = 0$, $p := \operatorname{char} k \Rightarrow p > 0$, und $\varphi \in \operatorname{End}(k)$ der Frobeniusendomorphismus. Bekanntlicherweise ist φ injektiv; weil k endlich ist, ist φ sogar ein Automorphismus. Daher gilt

$$f' = 0 \Leftrightarrow \exists\, a_i \in k : f(x) = \sum_i a_i x^{pi} = \sum_i \left(\varphi^{-1}(a_i) x^i\right)^p = \left(\sum_i \varphi^{-1}(a_i) x^i\right)^p$$

$\implies f$ ist nicht quadratfrei, insbesondere also kein mögliches Minimalpolynom. ▧

Die einzigen Körper, wo wir Pech haben könnten, sind also algebraische Erweiterungen von unendlichen Körpern mit Charakteristik $p > 0$, und zwar im Wesentlichen darum, weil die p-te Wurzel aus solchen Körpern herausführen kann.

In der Praxis wird man aber immer mit $\mathbb{Q}$ oder $\mathbb{F}_p$ als Grundkörper starten, obendrein sind algebraische Körpererweiterungen von unendlichen Körpern positiver Charakteristik in Singular (zur Zeit zumindest) nicht implementiert.

Die Existenz eines primitiven Elementes garantiert uns für die beiden obengenannten Fälle folgender Satz (der für unsere Zwecke noch etwas verschärft wurde):

Satz 3.29 (Primitives Element) *Sei $L = k(\alpha_1, \ldots, \alpha_n)$ eine separable Erweiterung eines beliebigen Körpers k. Dann gibt es eine Menge $W \subseteq k^{n-1}$, deren kanonische Projektion auf jede beliebige Komponente das Komplement einer endlichen Teilmenge von k ist, mit der Eigenschaft*

$$\forall\, \eta = (\eta_1, \ldots, \eta_{n-1}) \in W \ \text{gilt:} \quad k\left[\alpha_n + \sum_{i=1}^{n-1} \eta_i \alpha_i\right] = L$$

Insbesondere ist W nicht leer, falls k unendlich viele Elemente hat. Und im Fall $n = 2$ gilt: Bis auf endlich viele Ausnahmen ist für jedes $\eta \in k$: $k[\alpha, \beta] = k[\eta\alpha + \beta]$.

Dazu erst noch eine kurze **Bemerkung:** Ist α algebraisch über k, so ist $k(\alpha) = k[\alpha]$, denn wenn $\xi \in k(\alpha) \Rightarrow \exists\, f, g \in k[x] : g(\alpha) \neq 0,\ \xi = f(\alpha)/g(\alpha)$. Sei $h \in k[x]$ das Minimalpolynom von $\alpha \Rightarrow \operatorname{ggT}(g, h) = 1$, da h irreduzibel, $h(\alpha) = 0 \neq g(\alpha)$ und $\operatorname{ggT}(g, h) \mid h$.
$\Rightarrow \exists\, l, m \in k[x] : 1 = lg + mh \Rightarrow 1 = l(\alpha)g(\alpha) + m(\alpha)\underbrace{h(\alpha)}_{=0} \Rightarrow \xi = f(\alpha)l(\alpha)$

Beweis (vom Satz): Wir betrachten den Fall $n = 2$: Sei $L = k[\alpha, \beta]$, seien $f, g \in k[x]$ die Minimalpolynome von α bzw. β und $\overline{k}$ der algebraische Abschluß von k. Die Nullstellen von f bzw. g in $\overline{k}$ seien mit $\alpha_1{=}\alpha, \dots, \alpha_r$ bzw. $\beta_1{=}\beta, \dots, \beta_m$ bezeichnet, wobei die α_i paarweise verschieden sein sollen, ebenso die β_i. Definiere

$$W := k \setminus \left\{ \frac{\alpha - \alpha_i}{\beta_j - \beta} \;\middle|\; 1 \le i \le r,\ 1 < j \le m \right\}$$

Dann ist W das Komplement einer endlichen Menge und – falls k unendlich viele Elemente enthält – insbesondere nicht leer.[8]

Behauptung: $\quad k[\alpha + \eta\beta] = k[\alpha, \beta] \quad \forall\, \eta \in W$
Dazu genügt es zu zeigen, daß $\beta \in k[\alpha + \eta\beta]$. Sei also $\eta \in W$ beliebig gewählt, definiere $\xi := \alpha + \eta\beta$. Dann gilt: $g(\beta) = 0$ nach Wahl von g, und $f(\xi - \eta\beta) = f(\alpha) = 0$. $\quad f(\xi - \eta x), g(x) \in k(\xi)[x]$ sind also zwei Polynome mit β als gemeinsame Nullstelle. Dies ist auch die einzige gemeinsame Nullstelle, denn

- $g(x)$ hat die Nullstellen $\beta_1{=}\beta, \dots, \beta_m$
- $f(\xi - \eta x)$ hat die Nullstellen $\frac{\xi - \alpha_1}{\eta} = \beta, \dots, \frac{\xi - \alpha_r}{\eta} = \frac{\alpha - \alpha_r + \eta\beta}{\eta}$

Nun ist aber $\beta_j = \frac{\alpha - \alpha_i + \eta\beta}{\eta} \Leftrightarrow \eta(\beta_j - \beta) = \alpha - \alpha_i \Leftrightarrow \eta \notin W \ \vee\ (i, j) = (1, 1) \Leftrightarrow \beta_j = \beta = \frac{\alpha - \alpha_i + \eta\beta}{\eta}$, da ja $\eta \in W$ vorausgesetzt war.

Wenn aber β die einzige gemeinsame Nullstelle in $\overline{k}$ ist, dann ist der
$\mathrm{ggT}\big(f(\xi - \eta x), g(x)\big) = (x - \beta)^q$ für ein $q \in \mathbb{N}$. Da L nach Voraussetzung *separabel* ist, hat $g(x)$ nur einfache Nullstellen $\Rightarrow q = 1$. Aus $f(\xi - \eta x),\ g(x) \in k(\xi)[x]$ folgt $(x - \beta) = \mathrm{ggT}(f(\xi - \eta x), g(x)) \in k(\xi)[x]$ und daraus schließlich $\beta \in k(\xi)$, q.e.d.

Den Fall $n > 2$ beweist man mit Induktion:
Nach Induktionsvoraussetzung gibt es ein $V_1 \subseteq k^{n-2}$ „dicht" im Sinn der Behauptung, die wir für W zeigen wollen, so daß

$$k(\alpha_1, \dots, \alpha_n) = k\left[\alpha_{n-1} + \sum_{i=1}^{n-2} \eta_i \alpha_i\right][\alpha_n] \quad \forall\, \eta = (\eta_1, \dots, \eta_{n-2}) \in V_1$$

Im Fall $n = 2$ haben wir schon gesehen, daß es auch ein $V_2^\eta \subseteq k$ Komplement einer endlichen Menge gibt mit

$$k\left[\alpha_{n-1} + \sum_{i=1}^{n-2} \eta_i \alpha_i\right][\alpha_n] = k\left[\alpha_{n-1} + \sum_{i=1}^{n-2} \eta_i \alpha_i + \kappa\alpha_n\right] \quad \forall\, \kappa \in V_2^\eta$$

$$\text{Definiere also} \quad W := \bigcup_{\eta \in V_1} \{\eta\} \times V_2^\eta \ : \quad \text{dieses } W \text{ erfüllt den Satz.}$$

Denn die Reihenfolge der Indizierung der α_i spielt natürlich keine Rolle. ▧

[8] Man beachte ferner, daß die Menge $\left\{ \frac{\alpha - \alpha_i}{\beta_j - \beta} \mid 1 \le i \le r,\ 1 < j \le m \right\}$ im allg. keine Teilmenge von k ist wegen $\alpha_i, \beta_j \notin k$, d. h. die Anzahl der Elemente, die man aus k herausnehmen muß, ist normalerweise kleiner als die Mächtigkeit dieser Menge. Darum ist W auch bei endlichen Körpern „fast nie" leer; W ist in jedem Fall nicht leer, wenn k hinreichend viele Elemente hat, verglichen mit dem Grad der Minimalpolynome.

Korollar *Alle endlichen separablen Körpererweiterungen besitzen ein primitives Element.*

Beweis: Handelt es sich um einen Körper mit unendlich vielen Elementen, so ist auch der Grundkörper unendlich und die Behauptung folgt aus Satz 3.29, da $W \neq \emptyset$. Ist der Körper L endlich, so ist seine multiplikative Gruppe $L \setminus \{0\}$ zyklisch [Wae, §43]. Ein Erzeuger dieser Gruppe ist automatisch auch ein primitives Element. ▧

3.3.2 Berechnung des primitiven Elements

Nachdem gesichert ist, daß alle endlichen algebraischen Erweiterungen von Körpern der Charakteristik Null und von endlichen Körpern ein primitives Element besitzen, kümmern wir uns um dessen Berechnung. Die modifizierte Form von `HNΔ` liefert uns eine Folge von Körpererweiterungen $k = k_0 \subset k_1 \subset \ldots \subset k_n = \widetilde{k}$, die gegeben ist durch Polynome $f_1, \ldots, f_n$ mit folgender Eigenschaft:

$$f_i \in k_{i-1}[x] \quad \text{ist irreduzibel} \quad \text{und} \quad k_i = k_{i-1}[x]/\langle f_i \rangle$$

Wir bezeichnen mit $\quad k_i \ni \alpha_i := [x]_{f_i} \quad$ die Restklasse von x bezgl. f_i, d. h. α_i ist eine Nullstelle von f_i und f_i ist (nach eventueller Normierung) das Minimalpolynom von α_i. Damit können wir unser Problem folgendermaßen beschreiben:

Problem ($\mathfrak{P}$) :

Gegeben: ein Körper k sowie Polynome $g_i \in k[x_1, \ldots, x_i]$, durch die wiederum sukzessive algebraische Erweiterungen von k gegeben sind, und zwar wie folgt:

- $g_1 \in k[x_1]$ ist irreduzibel, und durch eine Nullstelle[9] α_1 von $g_1 =: f_{\alpha_1}$ im algebraischen Abschluß von k ist eine Körpererweiterung $k(\alpha_1) \supset k$ gegeben.
- Für $2 \leq i \leq n$ ist $f_{\alpha_i} := g(\alpha_1, \ldots, \alpha_{i-1}, x_i) \in k(\alpha_1, \ldots, \alpha_{i-1})[x_i]$ irreduzibel, und durch eine Nullstelle α_i dieses Polynoms ist eine Körpererweiterung $k(\alpha_1, \ldots, \alpha_i) \supset k(\alpha_1, \ldots, \alpha_{i-1})$ gegeben.

Insgesamt haben wir damit eine algebraische Erweiterung

$$L := k(\alpha_1, \ldots, \alpha_n) \supset k(\alpha_1, \ldots, \alpha_{n-1}) \supset \ldots \supset k(\alpha_1) \supset k \quad \text{definiert.}$$

Wir setzen außerdem voraus, daß $L \supset k$ separabel ist.

Gesucht: ein Minimalpolynom $f_\gamma \in k[x]$ für ein primitives Element γ von L.

[9] welche Nullstelle man wählt, spielt keine Rolle, denn die Körpererweiterungen $k(\beta)$ sind für alle Nullstellen β von f_{α_1} isomorph.

In einem ersten Verfahren, das praktisch immer zum Ziel führt, werden wir f_γ mit Hilfe von Standardbasen berechnen, wobei wir den Satz vom Primitiven Element benutzen und nur Linearkombinationen der α_i testen.

Sollte das keinen Erfolg haben (d. h. $W = \emptyset$ in Satz 3.29, was nur bei endlichen Körpern passieren kann), gibt es nach dem Korollar zum Satz trotzdem ein primitives Element. Wir sind dann allerdings gezwungen, alle Elemente von L zu testen. Die Berechnung des Minimalpolynoms wird dadurch schwieriger, aber es war mir möglich, eine Lösung zu finden, ebenfalls mit Standardbasen.

Zur Vorbereitung führen wir einige Begriffe aus der Computeralgebra ein und geben für uns wichtige Eigenschaften an. Dazu sei k ein Körper, $R = k[x_1, \ldots, x_n]$, $I \subset R$ ein Ideal und $<$ eine Ordnung auf der Menge der Monome von R.

Definitionen 3.16 Es bezeichne $\mathrm{L}(f)$ das *Leitmonom* von $f \in R$ bzgl. $<$ und $\mathrm{c}(f)$ den *Leitkoeffizient.* $\mathrm{L}(I) := \langle \mathrm{L}(f) \mid f \in I \rangle$ heißt das **Leitideal** von I.
$\{f_1, \ldots, f_m\} \subset R$ heißt **Standardbasis** von I, falls $\mathrm{L}(I) = \langle \mathrm{L}(f_1), \ldots, \mathrm{L}(f_m) \rangle$. Eine Standardbasis (SB) G heißt **minimal**, wenn $f, g \in G \Rightarrow \mathrm{L}(f) \nmid \mathrm{L}(g)$. G heißt **reduziert**, wenn gilt: $f \in G \Rightarrow$ kein Monom von f wird von einem $\mathrm{L}(g), g \in G$, geteilt, und wenn alle Elemente von G normiert sind (d. h. der Leitkoeffizient bzgl. $<$ ist 1). Eine reduzierte Standardbasis ist also insbesondere minimal.
Sei $\underline{x} = (x_1, \ldots, x_n)$, $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$ ein Multiindex, dann bezeichne $\underline{x}^\alpha := x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ ein Monom. Für $f, g \in k[\underline{x}]$ sei $\underline{x}^\alpha := \mathrm{kgV}(\mathrm{L}(f), \mathrm{L}(g))$. Definiere

$$\mathbf{spoly}(f,g) := \frac{\underline{x}^\alpha}{\mathrm{L}(f)} f - \frac{\mathrm{c}(f)}{\mathrm{c}(g)} \frac{\underline{x}^\alpha}{\mathrm{L}(g)} g$$

Falls $<$ eine Wohlordnung ist, definiere die **Normalform** $\mathrm{NF}(f \,|\, G)$ von $f \in R$ bzgl. $G = \{g_1, \ldots, g_m\} \subset R$ durch folgenden Algorithmus:

```
h = NF(f|G)
    h ← f
    while ∃ i: L(g_i) | L(h) do
          h ← spoly(h,g_i)
```

Satz 3.30 *Standardbasen haben folgende Eigenschaften (siehe z. B. [Cox, 2.§§5–7], [BW, Kap. 5.2 & 5.5]):*

- *Jedes Ideal* $(0) \neq I \subset R$ *besitzt (zu jeder Ordnung* $<$*) eine Standardbasis.*
- *Ist* $<$ *eine* Wohlordnung, *so ist eine Standardbasis von* I *sogar ein Erzeugendensystem von* I.
- *Die reduzierte Standardbasis von* I *ist* eindeutig.
- *Im Falle einer Wohlordnung ist* G *genau dann Standardbasis (von* $\langle G \rangle$*), wenn* $\mathrm{NF}(\mathrm{spoly}(f,g) \,|\, G) = 0 \quad \forall\, f, g \in G$

- ***Produktkriterium:*** $f, g \in k[\underline{x}]: \ \mathrm{L}(f) \cdot \mathrm{L}(g) = kgV(\mathrm{L}(f), \mathrm{L}(g))$
$\Longrightarrow \ \mathrm{NF}(\mathrm{spoly}(f,g) \,|\, \{f,g\}) = 0$

Die Berechnung des Minimalpolynoms f_γ beruht auf folgendem Satz:

Satz 3.31 *Sei k ein beliebiger Körper, seien $g_i \in k[x_1, \ldots, x_i]$, $\alpha_i \in \overline{k}$ und $f_{\alpha_i} \in k(\alpha_1, \ldots, \alpha_{i-1})[x_i]$ wie in der Beschreibung des Problems ($\mathfrak{P}$). Dann gilt:*

(i) $\mathfrak{M} := \langle g_1, g_2, \ldots, g_n \rangle \subset k[x_1, \ldots, x_n]$ *ist ein maximales Ideal, genauer:* $L := k(\alpha_1, \ldots, \alpha_n) \cong k[x_1, \ldots, x_n]/\mathfrak{M}$

(ii) Es gibt ein $W \subseteq k^{n-1}$ „dicht“ wie in Satz 3.29, so daß gilt:

$$\forall\, \xi \in W,\ y := x_n + \sum_{i=1}^{n-1} \xi_i x_i \quad \exists\, h_1, \ldots, h_n \in k[x]:$$
$$\mathfrak{M} = \langle x_1 - h_1(y), x_2 - h_2(y), \ldots, x_{n-1} - h_{n-1}(y), h_n(y) \rangle \tag{3.22}$$

und h_n ist irreduzibel.

Außerdem ist $\{x_1 - h_1, x_2 - h_2, \ldots, x_{n-1} - h_{n-1}, h_n\}$ eine Standardbasis bzgl. der Ordnung lex*: $x_1 > \ldots > x_{n-1} > x$.*

(iii) Falls $y = x_n + \sum_{i=1}^{n-1} \xi_i x_i$ und $h_i \in k[x]$ sind, so daß $\mathfrak{M}$ ein Erzeugendensystem wie in (3.22) hat, dann ist $\gamma := \alpha_n + \sum_{i=1}^{n-1} \xi_i \alpha_i$ ein primitives Element von L mit Minimalpolynom h_n (nach Normierung), und es gilt $\alpha_i = h_i(\gamma)$ für $i := 1, \ldots, n-1$.

Beweis: Für (i) verwenden wir vollständige Induktion:

$$\begin{aligned} k[x_1, \ldots, x_n]/\mathfrak{M} &= k[x_1, \ldots, x_n]/\langle f_{\alpha_1}, g_2, \ldots, g_n \rangle \\ &\cong k(\alpha_1)[x_2, \ldots, x_n]/\langle f_{\alpha_2}, g_3(\alpha_1, x_2, x_3), \ldots, g_n(\alpha_1, \ldots, x_n) \rangle \\ &\cong \ldots \cong k(\alpha_1, \ldots, \alpha_{n-1})[x_n]/\langle f_{\alpha_n} \rangle \cong k(\alpha_1, \ldots, \alpha_n) = L \end{aligned}$$

Wir können außerdem die Isomorphismen so wählen, daß wir als Komposition einen Isomorphismus von k-Algebren

$$\phi : k[x_1, \ldots, x_n]/\mathfrak{M} \xrightarrow{\cong} k(\alpha_1, \ldots, \alpha_n) \quad \text{erhalten mit} \quad \phi\left([x_i]_{\mathfrak{M}}\right) = \alpha_i$$

Zu (ii): Weil nach Annahme $L \supset k$ separabel ist, gibt es eine Menge $W \subset k^{n-1}$, mit deren Elementen sich nach Satz 3.29 aus den α_i ein primitives Element erzeugen läßt. Sei $\xi \in W$ ein beliebiges Element, $y := x_n + \sum_{i=1}^{n-1} \xi_i x_i$ und es bezeichne $\bar{x}_i, \bar{y}$ die Restklassen von x_i, y in $k[x_1, \ldots, x_n]/\mathfrak{M}$.

Weil $L \supset k$ endlich und $\bar{y} \in L$ ist, gibt es ein irreduzibles Polynom $h_n \in k[x]$ (z. B. das Minimalpolynom) mit $h_n(\bar{y}) = 0$. $h_1, \ldots, h_{n-1}$ konstruieren wir folgendermaßen:

$$\forall\, 1 \leq i \leq n-1: \ \phi(\bar{x}_i) = \alpha_i \in L \underbrace{=}_{\xi \in W} k\left[\alpha_n + \sum_{i=1}^{n-1} \xi_i \alpha_i\right] = \phi(k[\bar{y}])$$

$$\Longrightarrow \quad \bar{x}_i \in k[\bar{y}] \quad \Longrightarrow \quad \exists\, h_i \in k[x] : \bar{x}_i = h_i(\bar{y})$$

Nun ist aber $\quad h_i(\bar{y}) = \bar{x}_i \iff x_i - h_i(y) \in \mathfrak{M} \quad$ und $\quad h_n(\bar{y}) = 0 \iff h_n(y) \in \mathfrak{M}$.

$$\Longrightarrow \quad \mathfrak{N} := \langle x_1 - h_1(y), x_2 - h_2(y), \ldots, x_{n-1} - h_{n-1}(y), h_n(y)\rangle \subseteq \mathfrak{M}$$

$\mathfrak{N}$ ist ein maximales Ideal, denn $\quad k[x_1, \ldots, x_n]/\mathfrak{N} \cong k[x_n]/\langle h_n(x_n)\rangle \quad$ und h_n ist irreduzibel. Also folgt $\mathfrak{N} = \mathfrak{M}$.

Daß das Erzeugendensystem eine Standardbasis bzgl. *lex* ist, sieht man mit dem Produktkriterium, denn die Leitmonome der Erzeuger lauten $x_1, \ldots, x_{n-1}, x^{\deg h_n}$, sind also teilerfremd.

Zu (iii): Zunächst haben wir uns davon zu überzeugen, daß h_n irreduzibel ist. Das ist der Fall, denn $\mathfrak{M}$ ist maximal und

$$\begin{aligned} \psi:\ k[x_1, \ldots, x_n]/\mathfrak{M} &\cong k[x_1, \ldots, x_{n-1}, x]/\langle x_1 - h_1, \ldots, x_{n-1} - h_{n-1}, h_n\rangle \\ &\cong k[x]/\langle h_n\rangle \end{aligned}$$

$$\begin{array}{ccccc} [x_i] & \leftrightarrow & [x_i] = [h_i] & \leftrightarrow & [h_i] \\ [x_n] & \leftrightarrow & [x] - \sum_{i=1}^{n-1} \xi_i [x_i] & \leftrightarrow & [x] - \sum_{i=1}^{n-1} \xi_i [h_i] \end{array}$$

ist ein Isomorphismus. Für alle $1 \leq i \leq n$ ist

$$\psi \circ \phi^{-1}(h_i(\gamma)) = \psi\left(h_i\left([x_n] + \sum_{i=1}^{n-1} \xi_i [x_i]\right)\right) = h_i([x]) = [h_i]$$

(zur Definition von ϕ siehe Beweis von (i)). Speziell für $i = n$ erhalten wir $\psi \circ \phi^{-1}(h_n(\gamma)) = [h_n] = 0, \quad$ weswegen γ eine Nullstelle von h_n ist. Wenn man h_n noch normiert, erhält man also das Minimalpolynom von γ.

Sei $1 \leq i \leq n-1$, dann gilt $\qquad h_i(\gamma) \underset{\text{s.o.}}{=} \phi \circ \psi^{-1}([h_i]) = \phi([x_i]) = \alpha_i.$

Also sind $\alpha_1, \ldots, \alpha_{n-1} \in k(\gamma)$. Dann ist aber auch $\alpha_n = \gamma - \sum_{i=1}^{n-1} \xi_i \alpha_i \in k(\gamma)$, daher ist $k(\alpha_1, \ldots, \alpha_n) = k(\gamma)$. ▧

Algorithmus `Primitives Element`

Eingabe: $g_1, \ldots, g_n \ :\ g_i \in k[x_1, \ldots, x_i],\ g_i(\alpha_1, \ldots, \alpha_i) = 0,\ g_i(\alpha_1, \ldots, \alpha_{i-1}, x)$ irreduzibel in $k(\alpha_1, \ldots, \alpha_{i-1})[x]$

Ausgabe: $f_\gamma \in k[x_n]$ Minimalpolynom für ein primitives Element γ mit $k(\gamma) = k(\alpha_1, \ldots, \alpha_n)$

$h_1, \ldots, h_n \in k[x_n] :\ h_i(\gamma) = \alpha_i$

Schritt 1: Definiere $y := \sum_{i=1}^{n-1} a_i x_i + x_n$, wobei alle a_i per Zufallsgenerator gewählt werden.

Schritt 2: Berechne die reduzierte Standardbasis G von

$$\mathfrak{M} := \langle g_1(x_1), \ldots, g_{n-1}(x_1, \ldots, x_{n-1}), g_n(x_1, \ldots, x_{n-1}, y)\rangle$$

bezgl. *lex*: $x_1 > \ldots > x_{n-1} > x_n$

Schritt 3: Überprüfe, ob G die Eigenschaft von Satz 3.31 (ii) erfüllt. D. h.

- Hat G genau n Elemente?
- Sei $G =: \{f_1, \dots, f_n\}$. Gilt (nach evtl. Sortierung der Elemente) $\mathrm{L}(f_i) = x_i \quad \forall\, 1 \le i \le n-1$?

Wenn ja, dann hängen f_n und $f_i - x_i$, $1 \le i \le n-1$ nur noch von x_n ab (Definition von reduzierter SB).

3a Wenn G nicht in der Form vom Satz ist, dann gehe zurück zu Schritt 1, d. h. wähle die a_i neu.

3b Wenn G in der Form von Satz 3.31 ist, dann definiere $h_i := x_i - f_i$ für $i = 1, \dots, n-1$, $h_n := x_n + \sum_{i=1}^{n-1} a_i h_i$ und schließlich $f_\gamma := f_n$. Man beachte, daß diese Polynome alle univariat sind.
Gib f_γ und die h_i zurück.

Verifikation: Ist y durch die a_i vorgegeben, dann ist das $\mathfrak{M}$ in Schritt 2 das Bild von $\langle g_1, \dots, g_n\rangle =: \mathfrak{N}$ unter einem Automorphismus auf $k[x_1, \dots, x_n]$, der x_i, $i < n$ auf sich und x_n auf y abbildet. Sei $\{f_1, \dots, f_n\} \subset k[x_1, \dots, x_n]$ eine (reduzierte) Standardbasis. Genau dann ist $\{f_1, \dots, f_n\}$ ein Erzeugendensystem von $\mathfrak{M}$ (und damit (red.) SB von $\mathfrak{M}$), wenn

$$\left\{ f_1\left(x_1, \dots, x_{n-1}, x_n - \sum_{i=1}^{n-1} a_i x_i\right), \dots, f_n\left(x_1, \dots, x_{n-1}, x_n - \sum_{i=1}^{n-1} a_i x_i\right)\right\}$$

ein Erzeugendensystem von $\mathfrak{N}$ ist. Falls $(-a_1, \dots, -a_{n-1})$ ein Element von W aus Satz 3.31 (ii) ist, ist die Bedingung in Schritt 3 erfüllt. Falls $W \neq \emptyset$ terminiert der Algorithmus also, wenn k unendlich viele Elemente hat, terminiert er sogar fast sicher nach dem ersten Schleifendurchlauf. Falls k zu wenige Elemente hat und $W = \emptyset$ ist, ist dieser Algorithmus allerdings nicht geeignet. In diesem Fall sei auf den Algorithmus `Systematische Suche` im nächsten Abschnitt verwiesen. Obwohl bei endlichen Körpern der Erfolg nicht garantiert ist, lohnt es sich dennoch, in jedem Fall zuerst sein Glück mit `Primitives Element` zu versuchen, weil in der Praxis auch bei sehr kleinen Körpern W nicht leer ist. Nur muß man die Anzahl der Schleifendurchläufe bei endlichen Körpern auf einen sinnvollen Wert begrenzen; z. B. kann man bei kleinen Körpern statt zufällig gewählten a_i alle Elemente aus k^{n-1} testen (um zu vermeiden, daß die Schleife mehrmals mit dem gleichen Element durchlaufen wird) und die Suche abbrechen, wenn man damit fertig ist.

Nun zur Korrektheit der Ausgabe: Per Konstruktion ist

$$\mathfrak{M} = \langle x_1 - h_1, \dots, x_{n-1} - h_{n-1}, f_\gamma\rangle, \quad \text{was äquivalent ist zu}$$

$$\mathfrak{N} = \langle x_1 - h_1(z), \dots, x_{n-1} - h_{n-1}(z), f_\gamma(z)\rangle \quad \text{mit} \quad z := x_n - \sum_{i=1}^{n-1} a_i x_i$$

Mit Satz 3.31 (iii) folgt, daß f_γ Minimalpolynom von $\gamma := \alpha_n - \sum_{i=1}^{n-1} a_i \alpha_i$ ist (nach Voraussetzung ist f_γ bereits normiert), daß γ ein primitives Element ist und daß $\alpha_i = h_i(\gamma)$ ist für $1 \le i < n$. Daraus folgt schließlich auch, daß $\alpha_n = \gamma + \sum_{i=1}^{n-1} a_i \alpha_i = h_n(\gamma)$ ist. ▧

3.3.3 Primitives Element von endlichen Körpern

In diesem Teil beschränken wir uns darauf, das primitive Element einer durch zwei Elemente gegebenen Körpererweiterung $k(\alpha, \beta) \supset k$ zu berechnen.

Ist k unendlich, so ist W in Satz 3.31 nicht leer. Falls k endlich ist, hilft uns folgender Satz weiter:

Satz 3.32 *Sei k ein beliebiger Körper, $L = k(\alpha, \beta) \supset k$ eine algebraische Körpererweiterung, sei $f_\alpha \in k[x]$ Minimalpolynom von α und $f_\beta \in k[x,z]$ so, daß $f_\beta(\alpha, z) \in k(\alpha)[z]$ das Minimalpolynom von β ist.*

(i) Für beliebiges $g \in k[x,z]$ ist $\langle f_\alpha, f_\beta, g-y\rangle \cap k[x,y]$ ein maximales Ideal.

(ii) Falls es ein $g \in k[x,z]$ gibt mit $L = k\big(g(\alpha,\beta)\big)$, dann gibt es Polynome $h_1, h_2 \in k[y]$ mit folgenden Eigenschaften:

$$\mathfrak{M} := \langle f_\alpha, f_\beta, g-y\rangle \cap k[x,y] = \langle x - h_1, h_2\rangle \tag{3.23}$$

$$\begin{aligned}
&h_1(g(\alpha,\beta)) = \alpha, \quad h_2 \textit{ ist das Minimalpolynom von } g(\alpha,\beta),\\
&\deg h_2 = \deg f_\alpha \cdot \deg f_\beta(\alpha,z) \qquad \textit{und}\\
&\{x-h_1, h_2\} \quad \textit{ist eine Standardbasis von } \mathfrak{M} \textit{ bezgl. } \mathrm{lex}\textit{: } x > y.
\end{aligned}$$

(iii) Falls k ein endlicher *Körper ist und $g \in k[x,z]$ so, daß es $h_1, h_2 \in k[y]$ gibt mit*

$$\mathfrak{M} := \langle f_\alpha, f_\beta, g-y\rangle \cap k[x,y] = \langle x - h_1, h_2\rangle \tag{3.23}$$

und $\deg h_2 = \deg f_\alpha \cdot \deg f_\beta(\alpha, z)$, dann gilt:

h_2 ist irreduzibel in $k[y]$, $\gamma := g(\alpha,\beta)$ ist ein primitives Element von L und außerdem eine Nullstelle von h_2, $h_1(\gamma) = \alpha$.

Beweis: **Zu (i):**

$$\begin{array}{ccccccc}
k(\alpha,\beta) & \cong & k(\alpha)[z]/\langle f_\beta(\alpha,z)\rangle & \cong & k[x,z]/\langle f_\alpha, f_\beta\rangle & \cong & k[x,y,z]/\langle f_\alpha, f_\beta, g-y\rangle\\
\alpha & \longleftrightarrow & \alpha & \longleftrightarrow & [x] & \longleftrightarrow & [x]\\
\beta & \longleftrightarrow & [z] & \longleftrightarrow & [z] & \longleftrightarrow & [z]\\
 & & & & [g] & \longleftrightarrow & [y]=[g]
\end{array}$$

Daher ist $M := \langle f_\alpha, f_\beta, g-y\rangle \subset k[x,y,z]$ ein maximales Ideal, und es gibt einen Isomorphismus (von k-Algebren)

$$\varphi : k[x,y,z]/M \xrightarrow{\cong} k(\alpha,\beta)$$

mit $\varphi([x]) = \alpha$, $\varphi([z]) = \beta$ und $\varphi([y]) = \varphi([g]) = g(\varphi([x]), \varphi([z])) = g(\alpha,\beta)$.

Wir wollen nun zeigen, daß M auch ein maximales Ideal von $k[x,y]+M$ ist.
Seien $a_i \in k[x] : f_\beta = \sum_{i=0}^n a_i z^i$. $f_\beta(\alpha,z)$ ist ein Minimalpolynom $\Rightarrow \exists\, m \le n$:

$a_n(\alpha) = \ldots = a_{m+1}(\alpha) = 0,\ a_m(\alpha) = 1 \quad \Rightarrow\ f_\alpha$ ist ein Teiler von $a_n, \ldots, a_{m+1}$ und $a_m - 1 \quad \Rightarrow\ \exists\, q \in k[x,z] : \sum_{i=m}^{n} a_i z^i = q f_\alpha + z^m$

Darum ist z eine Nullstelle von

$$\begin{aligned} & f_\beta(x,u) - q(x,u) f_\alpha - f_\beta(x,z) + q(x,z) f_\alpha \\ = \;& \sum_{i=0}^{n} a_i u^i - \sum_{i=m}^{n} a_i u^i + u^m - f_\beta + q f_\alpha \\ = \;& u^m + \sum_{i=0}^{m-1} \underbrace{a_i}_{\in k[x]} u^i - \underbrace{f_\beta + q f_\alpha}_{\in M} \in (k[x,y] + M)[u] \end{aligned}$$

Aus diesem Grund ist z *ganz* über $k[x,y] + M$, woraus folgt, daß $\quad k[x,y,z] \supset k[x,y] + M \quad$ eine ganze Ringerweiterung ist.

Somit gilt für $\quad k[x,y,z] \supset k[x,y] + M \quad$ das Going-Up-Theorem [Ei, Prop. 4.15]. Darum folgt die Maximalität von $M \subset k[x,y] + M$ aus der Maximalität von $M \subset k[x,y,z]$.

Nach dem ersten Isomorphiesatz [Wae, §50] gibt es einen Isomorphismus

$$\psi : k[x,y]/\big(M \cap k[x,y]\big) \xrightarrow{\cong} \big(k[x,y] + M\big)/M\ , \quad [a] \mapsto [a]\ \forall\, a \in k[x,y]$$

Daraus folgt schließlich, daß auch $M \cap k[x,y]$ ein maximales Ideal in $k[x,y]$ ist.

Zu (ii): Sei $\gamma := g(\alpha,\beta)$ mit $L = k(\gamma)$. Weil γ algebraisch über k ist und wegen $\alpha \in L = k(\gamma) = k[\gamma]$ (Bemerkung zu Satz 3.29) gibt es ein Minimalpolynom $h_2 \in k[y]$ von γ und ein $h_1 \in k[y] : \alpha = h_1(\gamma)$.

Wir verwenden die im Beweis von Teil (i) eingeführten Begriffe φ, M weiter und stellen wegen der Existenz von φ fest, daß es einen Epimorphismus

$$\phi : k[x,y,z] \twoheadrightarrow k(\alpha,\beta)$$

gibt mit $\quad \phi(x) = \alpha, \quad \phi(z) = \beta, \quad \phi(y) = \gamma \quad$ und $\quad \operatorname{Ker}\phi = M$.

Weil γ ein primitives Element ist, ist auch die Einschränkung $\phi\big|_{k[x,y]+M}$ surjektiv,

$$\operatorname{Ker}\phi\big|_{k[x,y]+M} = (\operatorname{Ker}\phi) \cap (k[x,y] + M) = M \cap (k[x,y] + M) = M$$

daher gibt es einen Isomorphismus (von k-Algebren)

$$\Phi : (k[x,y] + M)/M \xrightarrow{\cong} k(\alpha,\beta)\ , \quad [x] \mapsto \alpha, \quad [y] \mapsto \gamma$$

$$\Phi([x - h_1]_M) = \Phi([x]_M) - h_1(\Phi[y]_M) = \alpha - h_1(\gamma) = 0 \quad \Longrightarrow \quad x - h_1 \in M$$

$$\Phi([h_2]_M) = h_2(\Phi([y]_M)) = h_2(\gamma) = 0 \quad \Longrightarrow \quad h_2 \in M$$

Offensichtlich gilt auch $\quad x - h_1, h_2 \in k[x,y], \quad$ woraus $\quad \langle x - h_1, h_2\rangle \subseteq M \cap k[x,y]$ folgt. $\quad \langle x - h_1, h_2\rangle \quad$ ist aber ebenfalls ein maximales Ideal, denn

$$k[x,y]/\left\langle x - h_1, h_2\right\rangle \cong k[y]/\left\langle h_2\right\rangle \cong k(\gamma) \quad \text{ist ein Körper,}$$

daher gilt $\langle x - h_1, h_2 \rangle = M \cap k[x,y] = \mathfrak{M}$.

Zum Beweis der letzten beiden Eigenschaften von h_1 und h_2:

$$\deg h_2 = [L : k] = [k(\alpha, \beta) : k(\alpha)] \cdot [k(\alpha) : k] = \deg f_\alpha \cdot \deg f_\beta(\alpha, z)$$

und daß $\{x - h_1, h_2\}$ eine Standardbasis ist, folgt wie im Beweis von Satz 3.31 aus dem Produktkriterium.

Zu (iii): Wegen (i) und (3.23) ist $\langle x - h_1, h_2 \rangle$ maximal und folglich $k[x,y]/\langle x - h_1, h_2 \rangle \cong k[y]/\langle h_2 \rangle$ ein Körper. Also ist $\langle h_2 \rangle \subset k[y]$ ein Primideal und h_2 somit irreduzibel.

Nach Voraussetzung gilt

$$[k[y]/\langle h_2 \rangle : k] = \deg h_2 = \deg f_\alpha \cdot \deg f_\beta(\alpha, z) = [k(\alpha) : k] \cdot [k(\alpha, \beta) : k(\alpha)] = [L : k]$$

Weil k endlich ist, folgt daß $k[y]/\langle h_2 \rangle$ und L *isomorph* sind (gleiche Anzahl Elemente). Es genügt also zu zeigen, daß γ eine Nullstelle von h_2 ist, denn dann ist γ zwangsläufig auch ein primitives Element.

Seien dazu wieder ψ und M wie im Beweis von (i) definiert und $\phi : k[x,y,z] \twoheadrightarrow k(\alpha, \beta)$ der im Beweis von (ii) eingeführte Epimorphismus. Anders als in (ii) wissen wir zwar noch nicht, ob $\phi\big|_{k[x,y]+M}$ surjektiv ist, aber nichtsdestotrotz folgt wie in (ii), daß $\operatorname{Ker} \phi\big|_{k[x,y]+M} = M$ ist, d. h.

$$\Phi : (k[x,y] + M)/M \longrightarrow k(\alpha, \beta)\,, \quad [x] \mapsto \alpha, \quad [y] \mapsto \gamma$$

existiert, ist k-linear und injektiv. Daraus können wir folgern:

$$h_2(\gamma) = h_2(\Phi([y]_M)) = \Phi([h_2(y)]_M) = \Phi(\psi([h_2]_{\mathfrak{M}})) = (\Phi \circ \psi)(0) = 0$$

Also ist γ ein primitives Element (und daher Φ auch hier wieder ein Isomorphismus). Schließlich gilt auch

$$h_1(\gamma) = \Phi([h_1]_M) = \Phi(\psi([h_1]_{\mathfrak{M}})) = \Phi(\psi([x]_{\mathfrak{M}})) = \Phi([x]_M) = \alpha$$

▧

Algorithmus `Systematische Suche`

Eingabe: $f_1 \in k[x]$ irreduzibel mit $f_1(\alpha) = 0$,
$f_2 \in k[x,z]$: $f_2(\alpha, z)$ irreduzibel in $k(\alpha)[z]$, $f_2(\alpha, \beta) = 0$.
Dabei sei $k \in \{\mathbb{F}_p \mid p \text{ Primzahl}\} \cup \{\mathbb{Q}\}$.

Ausgabe: $h_2 \in k[y]$ Minimalpolynom für ein primitives Element γ mit $k(\gamma) = k(\alpha, \beta)$,
$h_1 \in k[y]$ mit $h_1(\gamma) = \alpha$

Schritt 1: Falls char $k = p > 0$ ist: Mache das folgende für alle Polynome $g \in k[x,z]$ der Form

$$g = \sum_{i=0}^{\deg_x(f_1)-1} \sum_{j=0}^{\deg_z(f_2)-1} a_{ij}\, x^i z^j \;, \qquad a_{ij} := 0, 1, \ldots, p-1$$

Sonst (d. h. $k = \mathbb{Q}$) mache folgendes für alle Polynome $g \in k[x,z]$ der Form

$$g = ax + z \;, \qquad a := 0, 1, 2, \ldots$$

Schritt 2: Berechne die reduzierte Standardbasis G von

$$\mathfrak{M} := \langle f_1, f_2, g - y \rangle \cap k[x,y] \qquad \text{bezgl. } lex\text{: } x > y$$

Schritt 3: Überprüfe, ob G die Voraussetzung von Satz 3.32 (iii) erfüllt. D. h. ist $G = \{g_1, g_2\}$, wobei $L(g_1) = x$ und $\deg_y g_2 = \deg_x f_1 \cdot \deg_z f_2$ gilt?

Wenn nein, dann gehe zurück zu Schritt 1 und wähle das nächste g.
Wenn ja, dann definiere $h_1 := x - g_1$ und $h_2 := g_2$.
Weil G reduziert ist, sind $h_1, h_2 \in k[y]$. Gib h_1, h_2 zurück.

Verifikation: Falls $k = \mathbb{F}_p$ ist, folgt aus der Existenz eines primitiven Elements mit Satz 3.32 (ii) die Existenz eines $g \in k[x,z]$, welches die Voraussetzungen von 3.32 (iii) erfüllt. Weil $[k(\alpha) : k] = \deg f_1$ und $[k(\alpha,\beta) : k(\alpha)] = \deg f_2(\alpha, z) \le \deg_z f_2$ ist,[10] können wir zusätzlich voraussetzen, daß $\deg_x g < \deg_x f_1$ und $\deg_z g < \deg_z f_2$ ist. Dieses g wird nach endlich vielen Versuchen gefunden; Satz 3.32 (iii) beweist, daß die Ausgabe des Algorithmus für dieses g korrekt ist.

Falls $k = \mathbb{Q}$ ist, gibt es nach dem Satz vom Primitiven Element (Satz 3.29) nur endlich viele g der Form $g = ax + z$, $a \in \mathbb{Q}$, für die $g(\alpha,\beta)$ kein primitives Element ist. Es reicht darum völlig aus, diese linearen Polynome g zu testen mit $a \in \mathbb{N}$. Nach endlich vielen Versuchen finden wir ein g, welches die Voraussetzungen von Satz 3.32 (ii) erfüllt, also terminiert der Algorithmus auch für $k = \mathbb{Q}$.

Haben wir ein g gefunden, welches die Voraussetzungen von 3.32 (iii) (abgesehen von der Endlichkeit von k) erfüllt, so sehen wir im Beweis von (iii), daß alle Aussagen von 3.32 (iii) weiterhin gelten bis auf die, daß $g(\alpha,\beta)$ ein primitives Element ist. Diese Eigenschaft haben wir noch zu zeigen.

Es gilt $h_1(g(\alpha,\beta)) = \alpha \;\Rightarrow\; \alpha \in k(g(\alpha,\beta))$. Weil es ein $a \in \mathbb{N}$ gibt mit $g = ax + z$, ist auch $\beta = g(\alpha,\beta) - a\alpha \in k(g(\alpha,\beta))$. Also ist auch hier $g(\alpha,\beta)$ ein primitives Element. ▧

[10] Natürlich sollte man f_2 so wählen, daß $\deg_z f_2 = \deg f_2(\alpha, z)$ gilt, um die Anzahl der Versuche nicht unnötig zu erhöhen. Der Korrektheit tut das jedoch keinen Abbruch; da die Suche sowieso beim ersten gefundenen primitiven Element abgebrochen wird, wird sich die Wahl von f_2 außerdem kaum auf die Zahl der Versuche auswirken.

4 Berechnung von Kurveninvarianten aus der Hamburger-Noether-Entwicklung

Nun, da wir wissen, wie wir Hamburger-Noether-Entwicklungen berechnen können, beschäftigen wir uns noch damit, was sich mit ihnen weiter anfangen läßt. Folgende Informationen über eine irreduzible Kurve können wir bereits aus ihrer HNE gewinnen:

- Lemma 2.1 zeigt, wie wir aus der (unvollständigen) HNE (die Anfangsterme) eine(r) Parametrisierung durch Rückwärtseinsetzen gewinnen können. Dies wurde schon ausführlich in den Rechenbeispielen zum Algorithmus `Prim-Param-HNE` vorgeführt.

- Satz 2.6 gibt bereits einen Hinweis, welche Daten einer HNE relevant sind, wenn man Invarianten einer irreduziblen Kurve berechnen will. Es sind die Anzahl r der Zeilen der HNE, die Exponenten h_j und die $\underline{\nu}(z_j)$, $j = 1, \ldots, r$. Ferner enthält der Satz eine Vorschrift, wie man daraus die Multiplizitätensequenz berechnen kann.

- Außerdem können wir für reduzible Kurven aus dem Ergebnis von `HNE-reduzibel` sofort die Anzahl der Zweige ermitteln: Sie ist gleich der Anzahl der Blätter beider zurückgegebener Bäume (Satz 3.24).

Zunächst werden wir einige Formeln herleiten, wie man aus der Hamburger-Noether-Entwicklung einer irreduziblen Kurve ihre Invarianten berechnen kann. Für reduzible Kurven berechnen wir danach die Schnittmultiplizität zweier Zweige aus den dazugehörigen HNEs. Zum Schluß leiten wir aus den bereits gezeigten Formeln weitere Formeln für die Umrechnung zwischen verschiedenen Invarianten einer Kurve ab – dabei wird sich herausstellen, welche Invarianten ein vollständiges System bilden (d. h. aus welchen sich die Multiplizitätensequenz wieder eindeutig rekonstruieren läßt).

Generalvoraussetzung: Soweit nicht anders vermerkt sei in diesem Kapitel R der lokale Ring einer irreduziblen algebroiden Kurve mit maximalem Ideal $\mathfrak{m}$, $\{\mathrm{x}, \mathrm{y}\}$ sei ein Erzeugendensystem von $\mathfrak{m}$ und

$$
\begin{array}{rcl}
z_{-1} & = & a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{i-1} & = & a_{i,2}\, z_i^2 + a_{i,3}\, z_i^3 + \cdots\cdots + a_{i,h_i}\, z_i^{h_i} + z_i^{h_i} z_{i+1} \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{r-1} & = & a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots\cdots\cdots\cdots\cdots
\end{array} \qquad (\mathsf{HN})
$$

sei eine Hamburger-Noether-Entwicklung von R in der Basis $\{\mathrm{x}, \mathrm{y}\}$. Ferner vereinbaren wir

$$h_r := \infty, \quad a_{i,1} := 0 \text{ für } i := 1, \ldots, r, \qquad \text{so daß wir schreiben können:}$$

$$z_{i-1} = \sum_{j=1}^{h_i} a_{i,j} z_i^j + z_i^{h_i} z_{i+1} \quad \text{für } i := 0, 1, \ldots, r \qquad (\mathsf{HN}')$$

4.1 Die Multiplizitätensequenz

Zur Berechnung der Multiplizitätensequenz von R (wohlgemerkt eine irreduzible Kurve) brauchen wir nur noch eine Formel für $\underline{\nu}(z_j)$:

Lemma 4.1 *Mit den Voraussetzungen von* (HN) *definiere* $n_r := 1$ *und*

$$\forall\, 1 \le j \le r: \quad n_{j-1} := \begin{cases} n_j \cdot l & \textit{falls} \;\; \exists\, l \in \{2, 3, \ldots, h_j\} \\ & \quad \textit{minimal, so daß } a_{jl} \neq 0 \\ n_j \cdot h_j + n_{j+1} & \textit{falls} \;\; a_{jl} = 0 \;\; \forall\, 2 \le l \le h_j \end{cases}$$

Dann ist $\;n_j = \underline{\nu}(z_j)\;$ *für alle* $\;0 \le j \le r$.

Beweis: Damit die n_j wohldefiniert sind ist sicherzustellen, daß im Fall $1 \le r$ ein $l \ge 2$ existiert, für das $a_{rl} \neq 0$ ist. Wäre das nicht der Fall, folgte $z_{r-1} = 0$, was per definitionem von HNE für $r > 0$ nicht sein kann.

$n_r = 1 = \underline{\nu}(z_r)$ nach Korollar 1 zu Lemma 2.1.
Sei $1 \le j \le r$ und $n_i = \underline{\nu}(z_i)\;\; \forall\, j \le i \le r$. Es tritt einer der beiden folgenden Fälle ein:

a) $$z_{j-1} = \sum_{i=l}^{h_j} a_{ji} z_j^i + z_j^{h_j} z_{j+1}, \quad 2 \le l \le h_j, \quad a_{jl} \neq 0 \qquad \text{oder}$$

$$z_{j-1} = \sum_{i=l}^{\infty} a_{ji} z_j^i, \quad 2 \le l, \quad a_{jl} \neq 0$$

$$\underline{\nu}\left(a_{ji} z_j^i\right) = i\, \underline{\nu}(z_j), \text{ falls } a_{ji} \neq 0, \quad \underline{\nu}\left(z_j^{h_j} z_{j+1}\right) = h_j\, \underline{\nu}(z_j) + \underline{\nu}(z_{j+1}) > l\, \underline{\nu}(z_j)$$

$$\text{In beiden Fällen folgt} \quad \underline{\nu}(z_{j-1}) = l\, \underline{\nu}(z_j) = n_j \cdot l$$

b) $$z_{j-1} = z_j^{h_j} z_{j+1} \quad \Longrightarrow \quad \underline{\nu}(z_{j-1}) = h_j\, \underline{\nu}(z_j) + \underline{\nu}(z_{j+1}) = n_j \cdot h_j + n_{j+1} \qquad ▧$$

Ergebnis: Mit n_j definiert wie oben lautet die Multiplizitätensequenz von R

$$\underbrace{n_0,\ldots,n_0}_{h_0\text{-mal}},\underbrace{n_1,\ldots,n_1}_{h_1\text{-mal}},\ldots,\underbrace{n_{r-1},\ldots,n_{r-1}}_{h_{r-1}\text{-mal}},1\,(,1,1,1,\ldots) \tag{4.1}$$

Beweis: Satz 2.6 und Lemma 4.1 ▧

Diese Formel gilt auch für die Multiplizitätensequenzen der Zweige einer reduziblen Kurve. Um die Multiplizitätensequenz einer reduziblen Kurve berechnen zu können, fehlt uns noch eine geeignete Sortierung der Zweige und die Information, wann zwei Zweige beim Aufblasen getrennt werden. Letztere liefert uns später der Satz über den Zusammenhang zwischen Schnittmultiplizität und HNE (4.13) auf Seite 128 (siehe auch die anschließende Bemerkung auf Seite 130).

Jetzt ist der Zeitpunkt gekommen, um zu beweisen, daß jede irreduzible algebroide Kurve ein komplexes Modell besitzt, also eine algebroide Kurve über $\mathbb{C}$ mit der gleichen Multiplizitätensequenz.

Lemma 4.2 *Sei R der lokale Ring einer irreduziblen algebroiden Kurve über dem (beliebigen) algebraisch abgeschlossenen Körper k, sei*

$$z_{i-1} = \sum_{j=1}^{h_i} a_{ij} z_i^j + z_i^{h_i} z_{i+1}\,, \quad 0 \le i \le r \tag{HN$'$}$$

eine Hamburger-Noether-Entwicklung von R, und sei $\zeta : k \to \mathbb{C}$ irgendeine Abbildung mit der Eigenschaft $\zeta(x) = 0 \iff x = 0$.

Dann ist jede algebroide Kurve über $\mathbb{C}$, die eine Hamburger-Noether-Entwicklung

$$z'_{i-1} = \sum_{j=1}^{h_i} \zeta(a_{ij}) {z'_i}^j + {z'_i}^{h_i} z'_{i+1}\,, \quad 0 \le i \le r \tag{HN$'_{\mathbb{C}}$}$$

besitzt, ein komplexes Modell für R, und eine solche Kurve existiert auch.

Beweis: Die Existenz einer Kurve mit HNE (HN$'_{\mathbb{C}}$) folgt aus dem Korollar 2 zu Lemma 2.1. Per Konstruktion stimmen r und h_j bei (HN$'$) und (HN$'_{\mathbb{C}}$) überein, und wegen der Wahl von ζ liefert Lemma 4.1 auch die gleichen n_j. Die Behauptung folgt darum direkt aus (4.1). ▧

4.2 Geschlecht, charakteristische Exponenten und Puiseux-Paare

Dieser Abschnitt enthält Formeln, mit denen man das Geschlecht und die charakteristischen Exponenten direkt aus der Hamburger-Nother-Entwicklung berechnen kann. Die Puiseux-Paare wiederum können leicht aus den charakteristischen Exponenten ermittelt werden.

Definition 4.1 Sei wie in der Generalvoraussetzung (HN) eine HNE von R in der Basis $\{\mathrm{x},\mathrm{y}\}$ und $n_j = \underline{\nu}(z_j)$ (nach Lemma 4.1 auch direkt abzulesen aus (HN)).

Wir definieren $\boldsymbol{g, s_0, s_1, \ldots, s_g} \in \mathbb{N}$ durch folgende Bedingung:

$$0 = s_0 < s_1 < s_2 < \ldots < s_g \quad \text{und} \quad \big\{s_1, s_2, \ldots, s_g\big\} = \big\{j \;\big|\; 1 \le j \le r,\ n_j \,|\, n_{j-1}\big\} \tag{4.2}$$

Bemerkung: Wegen $1 \le n_{j+1} < n_j \ \forall\, 1 \le j \le r-1$ folgt aus Lemma 4.1: $n_j \,|\, n_{j-1} \iff \exists\, 2 \le l \le h_j : a_{jl} \ne 0$. Wegen $n_r = 1$ ist stets $s_g = r$.

Darum hat die HNE (HN) folgende Gestalt:

$$\begin{array}{rcl} z_{-1} & = & a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\ z_0 & = & z_1^{h_1} z_2 \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ z_{s_1-1} & = & a_{s_1,l_1}\, z_{s_1}^{l_1} + \cdots\cdots + a_{s_1,h_{s_1}}\, z_{s_1}^{h_{s_1}} + z_{s_1}^{h_{s_1}} z_{s_1+1} \\ z_{s_1} & = & z_{s_1+1}^{h_{s_1+1}} z_{s_1+2} \\ \cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ z_{s_g-1} & = & a_{s_g,l_g}\, z_{s_g}^{l_g} + \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \end{array} \tag{HN$_\mathrm{red}$}$$

mit $2 \le l_i \le h_{s_i}$, $a_{s_i,l_i} \ne 0$. Dieses Schema nennen wir eine Hamburger-Noether-Entwicklung **in reduzierter Form**.

Lemma 4.3 *g ist das Geschlecht der Kurve.*

Beweis: Aus der Definition des Geschlechtes (1.25) auf Seite 17 folgt mit Satz 2.6 sofort

$$g(R) = \sharp\left(\left\{\frac{n_0}{n_1}, \frac{n_1}{n_2}, \ldots, \frac{n_{r-1}}{n_r}\right\} \cap \mathbb{Z}\right) = \sharp\left(\left\{\frac{n_{s_1-1}}{n_{s_1}}, \ldots, \frac{n_{s_g-1}}{n_{s_g}}\right\}\right) = g$$

▧

Puiseux-Paare und charakteristische Exponenten lassen sich direkt ineinander umrechnen:

$$\text{Nach Definition 1.24 ist} \quad \frac{\beta_i}{\beta_0} = \frac{m_i}{n_1 n_2 \cdots n_i} \quad \text{und} \quad \mathrm{ggT}(m_i, n_i) = 1$$

Lemma 4.4 *Sind* $(m_1, n_1), (m_2, n_2), \ldots, (m_g, n_g)$ *die Puiseux-Paare und* $\beta_0, \ldots, \beta_g$ *die charakteristischen Exponenten von R, dann gilt:*

(i) $$\beta_0 = n_1 n_2 \cdots n_g$$

(ii) $$\beta_i = m_i\, n_{i+1}\, n_{i+2} \cdots n_g \quad \textit{für} \quad 1 \le i \le g$$

(iii) $$m_1 = \frac{\beta_1}{\mathrm{ggT}(\beta_0, \beta_1)}\,, \quad m_i = \frac{n_1 \cdots n_{i-1} \beta_i}{\mathrm{ggT}(\beta_0, n_1 \cdots n_{i-1} \beta_i)}\,, \quad 2 \le i \le g$$

$$n_1 = \frac{\beta_0}{\mathrm{ggT}(\beta_0, \beta_1)}\,, \quad n_i = \frac{\beta_0}{\mathrm{ggT}(\beta_0, n_1 \cdots n_{i-1} \beta_i)}\,, \quad 2 \le i \le g$$

Beweis: (i) $$\frac{\beta_1}{\beta_0} = \frac{m_1}{n_1}, \quad \mathrm{ggT}(m_1, n_1) = 1 \quad \Longrightarrow \quad n_1 \,|\, \beta_0$$

Induktionsannahme: $\exists\, 1 \leq i < g : n_1 n_2 \cdots n_i \,|\, \beta_0$

$$\frac{\beta_{i+1}}{\beta_0} = \frac{m_{i+1}}{n_1 \cdots n_{i+1}} \quad \Longrightarrow \quad \frac{\beta_{i+1}}{\frac{\beta_0}{n_1 \cdots n_i}} = \frac{m_{i+1}}{n_{i+1}}$$

$$\frac{\beta_0}{n_1 \cdots n_i} \in \mathbb{Z}, \quad \mathrm{ggT}(m_{i+1}, n_{i+1}) = 1 \quad \Longrightarrow \quad n_{i+1} \,\Big|\, \frac{\beta_0}{n_1 \cdots n_i}$$

$$\Longrightarrow \quad n_1 \cdots n_i n_{i+1} \,|\, \beta_0$$

Damit haben wir gezeigt: $\quad n_1 n_2 \cdots n_g \,|\, \beta_0, \quad$ woraus weiter folgt:

$$\beta_i = \beta_0 \frac{m_i}{n_1 n_2 \cdots n_i} = \underbrace{\frac{\beta_0}{n_1 \cdots n_g}}_{\in \mathbb{Z}} m_i n_{i+1} \cdots n_g$$

$$\begin{aligned} \Longrightarrow \quad 1 &= \mathrm{ggT}(\beta_0, \beta_1, \ldots, \beta_g) \\ &= \frac{\beta_0}{n_1 \cdots n_g} \cdot \mathrm{ggT}(n_1 \cdots n_g,\ m_1 n_2 \cdots n_g,\ \ldots,\ m_{g-1} n_g,\ m_g) \end{aligned}$$

$$\Longrightarrow \quad \frac{\beta_0}{n_1 \cdots n_g} = 1$$

(ii) folgt unmittelbar aus (i) und der Definition 1.24,
(iii) ist eine direkte Konsequenz aus $\frac{m_i}{n_i} = \frac{n_1 \cdots n_{i-1} \beta_i}{\beta_0}$ und $\mathrm{ggT}(m_i, n_i) = 1$. ▧

Um eine Formel für die charakteristischen Exponenten aufstellen zu können, untersuchen wir den Zusammenhang zwischen den charakteristischen Exponenten einer Kurve und denen ihrer quadratischen Transformierten. Zur Vorbereitung benötigen wir folgendes

Lemma 4.5 *Sei S der lokale Ring einer irreduziblen ebenen Kurve über einem algebraisch abgeschlossenen Körper der* Charakteristik Null, $\{\mathrm{x}, \mathrm{y}\}$ *ein Erzeuger des maximalen Ideals, sei*

$$x = t^n, \qquad y = \sum_{i=n}^{\infty} a_i\, t^i \tag{4.3}$$

eine primitive parametrische Darstellung von S in der Basis (x, y) *in Puiseux-Form (vgl. Satz 1.5). Mit Hilfe dieser Parametrisierung sind die charakteristischen Exponenten $\beta_0, \ldots, \beta_g$ von S definiert.*

Es gibt aber noch eine andere parametrische Darstellung von S in der gleichen Basis $\{\mathrm{x}, \mathrm{y}\}$ *in Puiseux-Form, nämlich*

$$x = \sum_{i=n}^{\infty} b_i\, t'^{\,i}, \qquad y = t'^{\,m} \tag{4.4}$$

wobei $m = \underline{\nu}(\mathrm{y})$.

Analog zu $\beta_0, \dots, \beta_g$ lassen sich auch mit dieser Parametrisierung „charakteristische Exponenten" definieren (wir nennen sie zur Unterscheidung charakteristische Exponenten der Parametrisierung *(4.4)):*

$$\beta'_0 := m, \quad \beta'_1 := \min\left\{ i \mid b_i \neq 0,\ \mathrm{ggT}(i, \beta'_0) < \beta'_0 \right\}$$

und falls $\nu > 1$ und $\mathrm{ggT}(\beta'_0, \dots, \beta'_\nu) > 1$

$$\beta'_{\nu+1} := \min\{ i \,|\, b_i \neq 0,\ \mathrm{ggT}(\beta'_0, \dots, \beta'_\nu, i) < \mathrm{ggT}(\beta'_0, \dots, \beta'_\nu)\}$$

Sei $g' \in \mathbb{N} : \mathrm{ggT}(\beta'_0, \dots, \beta'_{g'}) = 1$. ***Dann gilt:***

(a) Falls $n < m < \beta_1$, dann ist

$$g' = g + 1, \quad \beta'_1 = n \quad \text{und} \quad \beta'_{\nu+1} = \beta_\nu + n - m, \quad 1 \leq \nu \leq g$$

(b) Falls $n = m$ oder $m = \beta_1$, dann ist

$$g' = g \quad \text{und} \quad \beta'_\nu = \beta_\nu + n - m, \quad 1 \leq \nu \leq g$$

Beweis: Das ist eine Folgerung aus der *Inversionsformel von Abhyankar.*

Abhyankar definiert charakteristische Paare $(m_1, n_1), \dots, (m_u, n_u)$ für (4.3) und $(m'_1, n'_1), \dots, (m'_{u'}, n'_{u'})$ für (4.4) durch

$$y = \left(\sum_{j=1}^{u-1} \sum_{i=0}^{s_j} a_{ji}\, t^{(m_j+i)n_{j+1}\cdots n_u} \right) + \sum_{i=0}^{\infty} a_{ui}\, t^{m_u+i}$$

$$x = \left(\sum_{j=1}^{u'-1} \sum_{i=0}^{s'_j} b_{ji}\, t'^{\,(m'_j+i)n'_{j+1}\cdots n'_{u'}} \right) + \sum_{i=0}^{\infty} b_{u'i}\, t'^{\,m'_{u'}+i}$$

mit $\mathrm{ggT}(m_j, n_j) = 1, \quad a_{j0} \neq 0 \quad$ für $1 \leq j \leq u$,
$n_j > 1, \quad m_{j-1}/(n_1 \cdots n_{j-1}) < m_j/(n_1 \cdots n_j) \quad$ für $1 < j \leq u$
und s_j ist die größte ganze Zahl kleiner oder gleich $(m_{j+1}/n_{j+1}) - m_j$ für $1 \leq j < u$. Ferner soll $n_1 n_2 \cdots n_u = n$ sein. Analoge Bedingungen gelten für m'_j, n'_j, b_{j0}.

Die Inversionsformel [Ab, Theorem 1] von Abhyankar besagt dann folgendes:

$$u' = u, \quad m'_1 = n_1, \quad n'_1 = m_1,$$

$$m'_j = m_j - (m_1 - n_1) n_2 \cdots n_j \quad \text{und} \quad n'_j = n_j \quad \text{für} \quad 1 < j \leq u$$

Aus der Definition der charakteristischen Paare und Exponenten folgt ($\star$):

- Falls $n_1 > 1$, dann ist $\beta_i = m_i\, n_{i+1}\, n_{i+2} \cdots n_u \quad$ für $1 \leq i \leq u$ und $g = u$ (mit anderen Worten: $(m_1, n_1), \dots, (m_u, n_u)$ sind die Puiseux-Paare der Parametrisierung, vgl. Lemma 4.4(ii)).

- Falls $n_1 = 1$, dann ist $\beta_i = m_{i+1}\, n_{i+2}\, n_{i+3} \cdots n_u$ für $1 \le i \le u-1$ und $g = u - 1$
  (mit anderen Worten: $(m_2, n_2), \ldots, (m_u, n_u)$ sind die Puiseux-Paare der Parametrisierung).
- Analoges gilt für den Zusammenhang von (m'_i, n'_i) und β'_i.
- Stets ist $\beta_0 = n = n_1 n_2 \cdots n_u, \quad \beta'_0 = m = n'_1 n'_2 \cdots n'_{u'}$

Ferner gilt $n_1 = 1 \iff m \,|\, n,$ und $n'_1 = 1 \iff n \,|\, m.$

Nun zum Beweis des Lemmas: Falls $n < m < \beta_1 \Rightarrow \mathrm{ggT}(n, m) = n \Rightarrow n \,|\, m$. Außerdem gilt $m \nmid n$. Daher ist $g + 1 = u = u' = g'$.
Falls $n = m \Rightarrow n_1 = 1 = n'_1 \Rightarrow g + 1 = u = u' = g' + 1$.
Falls $m = \beta_1 \Rightarrow n \nmid m$, insbesondere also $n < m$. Daraus folgt $g = u = u' = g'$.

Der Zusammenhang von β_i und β'_i folgt durch Einsetzen der Inversionsformel in die jeweils richtige Gleichung von $(\star)$. ▧

Lemma 4.6 *Sei R_1 die quadratische Transformierte von R, $n := e(R)$, seien $\beta_0, \ldots, \beta_g$ bzw. $\beta_0^1, \ldots, \beta_{g_1}^1$ die charakteristischen Exponenten von R bzw. R_1.*

Wir definieren $h, n_1 \in \mathbb{N}$ durch $\beta_1 = hn + n_1, \quad 0 < n_1 < n$. (Man beachte $\beta_1 > \beta_0 = n \Rightarrow h \ge 1, \quad \mathrm{ggT}(\beta_0, \beta_1) < \beta_0 \Rightarrow n_1 > 0$.)
Dann gilt:

(a) $h > 1 \quad \Rightarrow g = g_1, \quad \beta_0 = \beta_0^1, \quad \beta_\nu^1 = \beta_\nu - n \quad \forall\, 1 \le \nu \le g$

(b) $h = 1, \quad n_1 \nmid n \Rightarrow g = g_1, \quad \beta_0^1 = n_1, \quad \beta_\nu^1 = \beta_\nu - n_1 \quad \forall\, 1 \le \nu \le g$

(c) $h = 1, \quad n_1 \,|\, n \Rightarrow g = g_1 + 1, \quad \beta_0^1 = n_1, \quad \beta_\nu^1 = \beta_{\nu+1} - n_1 \quad \forall\, 1 \le \nu \le g - 1$

Beweis: Ist $R_{\mathbb{C}}$ ein komplexes Modell für R, dann ist auch $(R_{\mathbb{C}})_1$ ein komplexes Modell für R_1. Wir können daher o.B.d.A. annehmen, daß R eine Kurve über dem Körper $\mathbb{C}$ ist. Dann besitzt R eine Puiseux-Parametrisierung

$$x = t^n, \qquad y = \sum_{i=n}^{\infty} a_i\, t^i$$

Aus Lemma 1.3 folgt, daß

$$x_1 = t^n, \qquad y_1 = \sum_{i=1}^{\infty} a_{n+i}\, t^i \tag{$*$}$$

eine Parametrisierung von R_1 ist.

Fall (a): $h > 1 \Rightarrow \beta_1 > 2n \Rightarrow a_{n+1} = \ldots = a_{2n-1} = 0$ (per def. von β_1).
Dann sind die charakteristischen Exponenten von $(*)$ bereits die charakteristischen Exponenten von R_1; es gilt $\beta_0 = n = e(R_1) = \beta_0^1$.

$$\text{Setze} \quad a'_i := a_{n+i} \quad \forall\, i \ge n, \quad \text{dann ist} \quad y_1 = \sum_{i=n}^{\infty} a'_i\, t^i$$

und es folgt aus $\operatorname{ggT}(n, \beta_1, \ldots, \beta_\nu, i-n) = \operatorname{ggT}(n, \beta_1, \ldots, \beta_\nu, i)$ mit Induktion:

$$\beta_1^1 = \beta_1 - n, \ \ldots, \ \beta_g^1 = \beta_g - n \quad \text{und} \quad \operatorname{ggT}(\beta_0, \ldots, \beta_\nu) = \operatorname{ggT}(\beta_0^1, \ldots, \beta_\nu^1)$$

weswegen schließlich auch $g_1 = g$ ist.

Falls $h = 1$ ist, dann ist $\beta_1 = n + n_1$ mit $0 < n_1 < n$

$$\Longrightarrow \quad y_1 = \sum_{i=n_1}^{\infty} a_{n+i}\, t^i = \sum_{i=n_1}^{\infty} a_i'\, t^i, \quad a_{n_1}' = a_{n+n_1} = a_{\beta_1} \neq 0$$

Wegen $n_1 < n$ müssen wir die Inversionsformel benutzen, um $\beta_1^1, \ldots, \beta_{g_1}^1$ zu berechnen. Es seien $\beta_0', \ldots, \beta_{g'}'$ die charakteristischen Exponenten von $(*)$. Per definitionem gilt:

$$\beta_0' = n, \quad \operatorname{ggT}(n, n_1) \leq n_1 < n \ \Rightarrow\ \beta_1' = n_1 = \beta_1 - n$$

und mit Induktion folgt für $1 \leq \nu < g$:

$$\operatorname{ggT}(\beta_0', \ldots, \beta_\nu', i) = \operatorname{ggT}(\beta_0, \ldots, \beta_\nu, i+n) \quad \Longrightarrow \quad \beta_{\nu+1}' = \beta_{\nu+1} - n$$

$$\Longrightarrow \quad \operatorname{ggT}(\beta_0', \beta_1', \ldots, \beta_{\nu+1}') = \operatorname{ggT}(\beta_0, \beta_1, \ldots, \beta_{\nu+1})$$

Daher ist auch $g' = g$.

Außer $(*)$ gibt es noch eine Puiseux-Parametrisierung von R_1:

$$x_1 = \sum_{i=n_1}^{\infty} b_i\, t'^{\,i}, \qquad y_1 = t'^{\,n_1}$$

mit $\operatorname{ord}_{t'}(x_1) = n$. Die charakteristischen Exponenten dieser Parametrisierung sind die gesuchten $\beta_0^1, \ldots, \beta_{g_1}^1$. Es gilt $\beta_0^1 = n_1 = e(R_1)$.

Fall (b): $n_1 \nmid n \ \Rightarrow\ n = \beta_1^1$. Mit Lemma 4.5 (b) folgt daraus

$$g_1 = g' = g \quad \text{und} \quad \beta_\nu - n = \beta_\nu' = \beta_\nu^1 + n_1 - n \quad \forall\, 1 \leq \nu \leq g$$

Fall (c): $n_1 \,|\, n \ \Rightarrow\ n_1 < n < \beta_1^1$. Mit Lemma 4.5 (a) folgt daraus

$$g_1 + 1 = g' = g, \quad \beta_{\nu+1} - n = \beta_{\nu+1}' = \beta_\nu^1 + n_1 - n \quad \forall\, 1 \leq \nu \leq g-1$$

Das war auch zu zeigen. ▨

Nun zur Formel für die charakteristischen Exponenten:

Satz 4.7 *Seien $n_j = \underline{\nu}(z_j)$ und $s_0, \ldots, s_g$ wie auf Seite 114 definiert, und seien $\beta_0, \ldots, \beta_{g'}$ die charakteristischen Exponenten von R. Dann gilt:*

(i) $\quad g = g'$

(ii) $$\beta_0 = n_0, \quad \beta_{\nu+1} = \sum_{j=0}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0, \quad 0 \leq \nu \leq g-1 \qquad (4.5)$$

Beweis: Mit Induktion über $M(R)$ (Anzahl der zur Glättung benötigten quadratischen Transformationen).

I.A.: $M(R) = 0 \Rightarrow n_0 = 1,\ r = 0 \Rightarrow \beta_0 = 1 \Rightarrow g' = 0 = g.$

I.V.: $M(R) > 0$, die Behauptung sei richtig für alle R' mit $M(R') < M(R)$. Ferner sei $(\mathsf{HN})_{\mathsf{red}}$ die HNE von R (in reduzierter Form) und R_1 die quadratische Transformierte von R mit charakteristischen Exponenten $\beta_0^1, \ldots, \beta_{g_1'}^1$ und Geschlecht g_1.

Wie in Lemma 4.6 betrachten wir $\beta_1 = hn + m, \quad 0 < m < n := e(R),$ und die drei auftretenden Fälle.
β_1 existiert, denn wegen $M(R) > 0$ ist $\beta_0 = n_0 > 1 \Rightarrow g' > 0$.

(a) $h > 1 \underset{4.6}{\Longrightarrow} g' = g_1', \quad \beta_0^1 = \beta_0, \quad \beta_\nu^1 = \beta_\nu - n, \quad 1 \le \nu \le g'$

Aus $e(R_1) = \beta_0^1 = \beta_0 = e(R)$ folgern wir, daß $h_0 > 1$ in $(\mathsf{HN})_{\mathsf{red}}$ gilt (Satz 2.5) und man die HNE von R_1 aus $(\mathsf{HN})_{\mathsf{red}}$ erhält, indem man die erste Zeile ersetzt durch

$$z_{-1} = a_{02}\, z_0 + a_{03}\, z_0^2 + \ldots + a_{0h_0}\, z_0^{h_0-1} + z_0^{h_0-1} z_1$$

Hieraus folgt $g = g_1 \underset{\mathrm{IV}}{=} g_1' = g'$ (Lemma 4.3) und

$$\begin{aligned} \beta_{\nu+1} &= \beta_{\nu+1}^1 + n \underset{\mathrm{IV}}{=} (h_0 - 1)n_0 + \sum_{j=1}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0 + n_0 \\ &= \sum_{j=0}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0 \end{aligned}$$

(b) $h = 1, \quad m \nmid n \underset{4.6}{\Longrightarrow} g' = g_1', \quad \beta_0^1 = m, \quad \beta_\nu^1 = \beta_\nu - m, \quad 1 \le \nu \le g'$

Aus $e(R_1) = m < n = e(R)$ folgern wir, daß $h_0 = 1$ in $(\mathsf{HN})_{\mathsf{red}}$ gilt (Satz 2.5) und man die HNE von R_1 aus $(\mathsf{HN})_{\mathsf{red}}$ erhält, indem man die erste Zeile wegläßt.

Dadurch verschiebt sich die Indizierung der Größen, aus denen die charakteristischen Exponenten berechnet werden; sei

$$z_{i-1}' = \sum_{j=1}^{h_i'} a_{ij}'\, z_i'^{\,j} + z_i'^{\,h_i'} z_{i+1}', \quad 0 \le i \le r'$$

eine HNE von R_1, dann ist

$$r' = r - 1, \quad h_i' = h_{i+1}, \quad n_i' = n_{i+1} \quad \forall\, 0 \le i \le r - 1$$

Daraus erhalten wir für die reduzierte Form dieser HNE

$$s_0' = 0, \quad s_\nu' = s_\nu - 1 \quad \forall\, 1 \le \nu \le g$$

denn wegen $m = \beta_0^1 = e(R_1) = n_0' = n_1$ gilt $n_1 \nmid n_0 \Rightarrow s_1 > 1$.
Das bedeutet schließlich auch $g_1 = g$.

Nun können wir Induktion anwenden: $g = g_1 \underset{\mathrm{IV}}{=} g'_1 = g'$ und $\forall\, 1 \le \nu \le g-1$:

$$\begin{aligned}
\beta_{\nu+1} &= \beta^1_{\nu+1} + m \underset{\mathrm{IV}}{=} \sum_{j=0}^{s'_\nu} h'_j n'_j + n'_{s'_\nu} + n'_{s'_\nu+1} - n'_0 + m \\
&= \sum_{j=0}^{s_\nu - 1} h_{j+1} n_{j+1} + n_{s_\nu} + n_{s_\nu+1} - n_1 + n_1 \\
&= \sum_{j=1}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} + h_0 n_0 - n_0 = \sum_{j=0}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0
\end{aligned}$$

Den Fall $\nu = 0$ behandeln wir gesondert:

$$\beta_1 = hn + m = h_0 n_0 + n_1 = \sum_{j=0}^{s_0} h_j n_j + n_{s_0} + n_{s_0+1} - n_0$$

(c) $h = 1, \quad m \,|\, n$

$$\underset{4.6}{\Longrightarrow} \quad g' = g'_1 + 1, \quad \beta^1_0 = m, \quad \beta^1_\nu = \beta_{\nu+1} - m, \quad 1 \le \nu \le g' - 1$$

Wie im Fall (b) folgt $h_0 = 1$ und weiter

$$r' = r - 1, \quad h'_i = h_{i+1}, \quad n'_i = n_{i+1} \quad \forall\, 0 \le i \le r-1; \quad \text{ferner} \quad m = n_1$$

Andererseits gilt jetzt $n_1 \,|\, n_0 \quad \Rightarrow \quad s_1 = 1.$ Daher ist

$$s_1 - 1 = 0 = s'_0 \quad \Longrightarrow \quad s'_\nu = s_{\nu+1} - 1 \ \text{ für } \ 0 \le \nu \le g-1 \quad \Longrightarrow \quad g_1 = g - 1$$

Daraus resultiert zum dritten Mal die Behauptung: $g = g_1 + 1 \underset{\mathrm{IV}}{=} g'_1 + 1 = g'$ und $\forall\, 1 \le \nu \le g-1$:

$$\begin{aligned}
\beta_{\nu+1} &= \beta^1_\nu + m \underset{\mathrm{IV}}{=} \sum_{j=0}^{s'_{\nu-1}} h'_j n'_j + n'_{s'_{\nu-1}} + n'_{s'_{\nu-1}+1} - n'_0 + m \\
&= \sum_{j=1}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_1 + n_1 \\
&= \sum_{j=0}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0
\end{aligned}$$

Die Richtigkeit der Formel für β_1 folgt genauso wie im Fall (b) – in den Beweis geht nur $h_0 = 1 = h$, $n = n_0$ und $m = n_1$ ein. ▨

4.3 Wertehalbgruppe, Grad des Konduktors und δ-Invariante

Campillo hat gezeigt, daß die Erzeuger der Wertehalbgruppe ein vollständiges System von Invarianten für die Gleichauflösung sind [Ca, 4.3.11]. Dazu hat er

$\overline{\beta}_0, \ldots, \overline{\beta}_g$ zuerst anders definiert („Werte maximalen Kontaktes vom Geschlecht i"), dann gezeigt, daß diese ein vollständiges System von Invarianten sind [Ca, 4.3.7] und schließlich bewiesen, daß $\overline{\beta_0}, \ldots, \overline{\beta_g}$ identisch mit dem minimalen Erzeugendensystem der Wertehalbgruppe sind [Ca, 4.3.10].

Das bedeutet, daß die Wertehalbgruppe einer Kurve mit der Wertehalbgruppe ihres komplexen Modells übereinstimmt. Die Formel für die Berechnung der Wertehalbgruppe in Charakteristik 0 findet sich bei Zariski [Za2]. Ihre Herleitung stellen wir im folgenden dar:

Lemma 4.8 *Sei R der lokale Ring einer singulären irreduziblen algebroiden Kurve über einem algebraisch abgeschlossenen Körper K der Charakteristik* 0. *Dann gibt es ein Erzeugendensystem* $\{\mathrm{x}, \mathrm{y}\}$ *des maximalen Ideals von R und eine primitive parametrische Darstellung*

$$x(t) = t^n, \qquad y(t) = \sum_{i=m}^{\infty} a_i\, t^i \quad \textit{mit} \quad m > n,\ n \nmid m$$

Bemerkung: Wollten wir nur die Halbgruppe *irgendeines* komplexen Modells ausrechnen, könnten wir sogar o.B.d.A. annehmen, daß

$$x(t) = t^{\beta_0}, \qquad y(t) = \sum_{i=1}^{g} a_{\beta_i}\, t^{\beta_i}$$

ist (denn diese Kurve hat offensichtlich die gleichen charakteristischen Exponenten wie die ursprüngliche Kurve), aber mit geringem Mehraufwand können wir so für Charakteristik 0 explizit mitbeweisen, wofür wir sonst auf [Ca] verwiesen haben – daß die Wertehalbgruppe allein von der Multiplizitätensequenz abhängt.

Beweis des Lemmas: Nach dem Satz von der Puiseuxentwicklung gibt es einen regulären Parameter $T \in \overline{R}$, $x(t), y(t) \in K[[t]]$:

$$x(t) = t^n, \qquad y(t) = \sum_{i=m}^{\infty} a_i\, t^i, \quad \text{mit} \quad m \geq n$$

und $\mathrm{x} := x(T)$, $\mathrm{y} := y(T)$ erzeugen das maximale Ideal. Falls es ein $h \in \mathbb{N}$ gibt mit $m = hn$, dann erzeugen auch x und $\mathrm{y} - a_m \mathrm{x}^h$ das maximale Ideal. Eine primitive parametrische Darstellung von R in dieser Basis ist

$$x(t) = t^n, \qquad y'(t) = a_{m+k}\, t^{m+k} + \ldots$$

wobei $k > 0$ minimal ist mit $a_{m+k} \neq 0$. Wenn $n \nmid m + k$ sind wir fertig, sonst iterieren wir das Ganze. Nach endlich vielen Schritten wird $m + k = \beta_1$ und wir sind fertig. ▧

Falls R eine glatte Kurve ist, gilt $R = \overline{R}$ und daher

$$\Gamma = \underline{\nu}(\overline{R} \setminus \{0\}) = \mathbb{N} = \langle 1 \rangle$$

Folglich ist $\{1\} = \{\beta_0\}$ ein minimales Erzeugendensystem der Wertehalbgruppe. Im folgenden nehmen wir daher an, daß R singulär ist und (x, y) eine Parametrisierung von R mit den Eigenschaften des vorausgehenden Lemmas.

Definition 4.2 Seien $\beta_0, \ldots, \beta_g$ die charakteristischen Exponenten von R, dann definiere

$$e_i := \mathrm{ggT}(\beta_0, \ldots, \beta_i), \quad 0 \leq i \leq g$$

Per definitionem der β_j gilt $\beta_0 = e_0 > e_1 > e_2 > \ldots > e_g = 1$ und $e_i \mid e_{i-1}$. Für $1 \leq i < g$ sei ferner $h_i \in \mathbb{N}$ maximal mit $\beta_i + h_i e_i < \beta_{i+1}$

Bemerkung: (i) Unser y läßt sich nun folgendermaßen schreiben:

$$y = b_1\, t^{\beta_1} + \sum_{0<j\leq h_1} a_{j1}\, t^{\beta_1 + je_1} + b_2\, t^{\beta_2} + \sum_{0<j\leq h_2} a_{j2}\, t^{\beta_2 + je_2} + \ldots + b_g\, t^{\beta_g} + \sum_{j=1}^{\infty} a_{jg}\, t^{\beta_g + j} \tag{4.6}$$

mit $b_i \neq 0 \quad \forall\, i$

(ii) Nach Lemma 4.4 gelten für die Puiseux-Paare $(m_1, n_1), \ldots, (m_g, n_g)$ von R folgende Beziehungen:

$$n = \beta_0 = n_1 \cdots n_g, \quad e_{i-1} = n_i n_{i+1} \cdots n_g \implies \beta_i = m_i e_i, \quad e_{i-1} = n_i e_i \tag{4.7}$$

Definition 4.3 Mit den oben eingeführten Bezeichnungen definieren wir

$$\begin{aligned} z_q &:= \sum_{m\leq i<\beta_q} a_i\, t^i \\ &= b_1\, t^{\beta_1} + \sum_{0<j\leq h_1} a_{j1}\, t^{\beta_1+je_1} + \ldots + b_{q-1}\, t^{\beta_{q-1}} + \sum_{0<j\leq h_{q-1}} a_{j,q-1}\, t^{\beta_{q-1}+je_{q-1}} \end{aligned}$$

Sei $m \leq i < \beta_q$ mit $a_i \neq 0$. Dann ist i teilbar durch e_{q-1} (aus 4.7).
Wir definieren $\rho_i := i/e_{q-1}$, dann ist

$$z_q = \sum_{m\leq i<\beta_q} a_i\, t^{\rho_i e_{q-1}} = \sum_{m\leq i<\beta_q} a_i\, x^{\frac{\rho_i}{n_1 n_2 \cdots n_{q-1}}}$$

Sei schließlich $\omega \in K$ eine n-te Einheitswurzel, $\varphi_\nu \in \mathrm{Aut}_K(K[\![t]\!])$ der durch $\varphi_\nu(t) = \omega^\nu t$ definierte Automorphismus und

$$f_q(Z) := \prod_{\nu=1}^{n_1\cdots n_{q-1}} (Z - \varphi_\nu(z_q))$$

Als ein Resultat aus der Galois-Theorie erhalten wir $f_q(Z) \in K[\![x]\!][Z]$ wegen $\varphi_\nu(f_q) = f_q \quad \forall\, \nu$.

Wir definieren $\quad \xi_q := f_q(\mathrm{y}) \in K[\![\mathrm{x}]\!][\mathrm{y}] \subset R, \quad 2 \leq q \leq g$

und $\quad \overline{\beta}_q := \underline{\nu}(\xi_q) \implies \overline{\beta}_q \in \Gamma \quad \forall\, 2 \leq q \leq g$

Sei außerdem $\overline{\beta}_0 := \underline{\nu}(\mathrm{x}) = n$ und $\overline{\beta}_1 := \underline{\nu}(\mathrm{y}) = \beta_1$. $\mathrm{x}, \mathrm{y} \in R \Rightarrow \overline{\beta}_0, \overline{\beta}_1 \in \Gamma$

Damit haben wir $g+1$ Elemente $\overline{\beta}_0, \overline{\beta}_1, \ldots, \overline{\beta}_g \in \Gamma$ der Wertehalbgruppe konstruiert. Es bleibt zu zeigen, daß diese $\overline{\beta}_i$ die Halbgruppe erzeugen, und daß sie ein minimales Erzeugendensystem bilden.

Satz 4.9

(i) Für alle $q \in \{2, \ldots, g\}$ gilt:

$$\overline{\beta}_q = (n_1 - 1)n_2 \cdots n_{q-1}\beta_1 + (n_2 - 1)n_3 \cdots n_{q-1}\beta_2 + \ldots + (n_{q-1} - 1)\beta_{q-1} + \beta_q \tag{4.8}$$

und daher auch

$$\overline{\beta}_q = n_{q-1}\overline{\beta}_{q-1} - \beta_{q-1} + \beta_q, \qquad \overline{\beta}_0 = \beta_0, \quad \overline{\beta}_1 = \beta_1 \tag{4.9}$$

(ii) Sei $\eta \in R$. Nach dem Weierstraßschen Vorbereitungssatz gibt es (genau) ein Polynom $\psi \in K[\![\mathrm{x}]\!][y]$ vom Grad $d < n$ mit $\psi(\mathrm{y}) = \eta$. Sei $0 \leq q \leq g$, so daß $n_1 \cdots n_q > d$ ist. Dann gilt:

$$\underline{\nu}(\eta) \in \sum_{\nu=0}^{q} \overline{\beta}_\nu \, \mathbb{Z}_+$$

Insbesondere gilt also $\langle \overline{\beta}_0, \ldots, \overline{\beta}_g \rangle = \Gamma$.

(iii) $$\forall\, 1 \leq q \leq g: \quad \mathrm{ggT}(e_{q-1}, \overline{\beta}_q) = e_q$$

Außerdem ist $\overline{\beta}_q$ das kleinste Element von Γ, welches nicht durch e_{q-1} teilbar ist, und daher das kleinste Element aus $\Gamma \setminus \langle \overline{\beta}_0, \ldots, \overline{\beta}_{q-1} \rangle$. Das bedeutet, daß das Erzeugendensystem der $\overline{\beta}_i$ auch minimal ist.

Beweis: (i): Per definitionem ist

$$\overline{\beta}_q = \mathrm{ord}_t \left(\prod_{\nu=1}^{n_1 \cdots n_{q-1}} (y - \varphi_\nu(z_q)) \right) = \sum_{\nu=1}^{n_1 \cdots n_{q-1}} \mathrm{ord}_t \, (y - \varphi_\nu(z_q))$$

$\varphi_\nu\left(t^{\beta_1}\right) = \omega^{\nu\beta_1} t^{\beta_1}$, daher gilt $\varphi_\nu\left(t^{\beta_1}\right) = t^{\beta_1} \iff n \mid \nu\beta_1 \iff n_1 \mid \nu$,
denn $\beta_1 = m_1 e_1$, $n = n_1 e_1$ und $\mathrm{ggT}(m_1, n_1) = 1$ (siehe 4.7).

Es folgt $\mathrm{ord}(y - \varphi_\nu(z_q)) = \beta_1 \iff n_1 \nmid \nu$.
Falls $n_1 \mid \nu \implies \varphi_\nu(t^{e_1}) = t^{e_1} \Rightarrow \mathrm{ord}(y - \varphi_\nu(z_q)) \geq \beta_2$
(alle Terme von kleinerem Grad heben sich auf).

Es gibt also genau $n_1 n_2 \cdots n_{q-1} - n_2 n_3 \cdots n_{q-1}$ Faktoren von $f_q(y)$ mit $\mathrm{ord}_t(y - \varphi_\nu(z_q)) = \beta_1$. Daher ist

$$\overline{\beta}_q = (n_1 - 1)n_2 \cdots n_{q-1}\beta_1 + \sum_{\nu=1}^{n_2 \cdots n_{q-1}} \mathrm{ord}_t(y - \varphi_{n_1\nu}(z_q))$$

Analog folgern wir, daß $\quad \varphi_{n_1\nu}(t^{\beta_2}) = t^{\beta_2} \iff n_2 \nmid \nu \quad$ gilt
und $\quad n_2 \mid \nu \implies \mathrm{ord}_t(y - \varphi_{n_1\nu}(z_q)) \geq \beta_3$
woraus wieder folgt:

$$\sum_{\nu=1}^{n_2\cdots n_{q-1}} \mathrm{ord}(y - \varphi_{n_1\nu}(z_q)) = (n_2-1)n_3\cdots n_{q-1}\beta_2 + \sum_{\nu=1}^{n_3\cdots n_{q-1}} \mathrm{ord}(y - \varphi_{n_1 n_2\nu}(z_q))$$

Nach $q-1$ Schritten erhalten wir

$$\overline{\beta}_q = (n_1-1)n_2\cdots n_{q-1}\beta_1 + \ldots + (n_{q-1}-1)\beta_{q-1} + \mathrm{ord}(y - \varphi_{n_1\cdots n_{q-1}}(z_q))$$

Wegen $\varphi_{n_1\cdots n_{q-1}}(z_q) = z_q$ ist $\mathrm{ord}(y - \varphi_{n_1\cdots n_{q-1}}(z_q)) = \beta_q$, woraus (4.8) folgt.

Die Formel (4.9) folgt trivial aus (4.8) bzw. ist von der Definition von $\overline{\beta}_0, \overline{\beta}_1$ her bereits klar.

(ii) zeigen wir mit Induktion über q:

I.A.: $q = 0 \Rightarrow d < 1 \Rightarrow d = 0 \Rightarrow \psi$ ist konstant $\Rightarrow \eta \in K[[\mathrm{x}]] \Rightarrow \overline{\beta}_0 \mid \underline{\nu}(\eta) \Rightarrow \underline{\nu}(\eta) \in \overline{\beta}_0\mathbb{Z}_+$

I.V.: Die Behauptung (ii) sei richtig für $q-1$.

I.S.: $\deg f_q = n_1\cdots n_{q-1} \Rightarrow \exists\, s < n_q,\ A_0,\ldots,A_s \in K[[\mathrm{x}]] : \deg A_i < n_1\cdots n_{q-1}$ und

$$\psi = A_0 + A_1 f_q + \ldots + A_s f_q^s \quad\implies\quad \eta = A_0(\mathrm{y}) + A_1(\mathrm{y})\xi_q + \ldots + A_s(\mathrm{y})\xi_q^s$$

Nach I.V. ist $\underline{\nu}(A_i(\mathrm{y})) \in \sum_{\nu=0}^{q-1} \overline{\beta}_\nu\,\mathbb{Z}_+$ für alle i. Insbesondere gilt wegen (4.8):

$$\underline{\nu}(A_i(\mathrm{y})) \equiv 0 \pmod{e_{q-1}} \quad\implies\quad \underline{\nu}(A_i(\mathrm{y})\xi_q^i) \equiv i\overline{\beta}_q \pmod{e_{q-1}}$$

Aus (4.8) folgt

$$\forall\, q : e_q \mid \overline{\beta_q} \underset{(4.9)}{\implies} \mathrm{ggT}(e_{q-1}, \overline{\beta}_q) = e_q$$

Das beweist den ersten Teil von (iii), und es folgt daraus wegen $e_{q-1} = n_q e_q$:

$$\forall\, 0 \leq i,j < n_q,\ i \neq j : (i-j)\overline{\beta}_q \not\equiv 0 \pmod{e_{q-1}}$$

Alle Summanden von η haben also einen unterschiedlichen Wert $\underline{\nu}(A_i(\mathrm{y})\xi_q^i)$, darum ist $\underline{\nu}(\eta)$ einer dieser Werte.

Damit ist bewiesen: $\quad \underline{\nu}(\eta) \in \sum_{\nu=0}^{q} \overline{\beta}_\nu\,\mathbb{Z}_+$

Zu (iii): $\mathrm{ggT}(e_{q-1}, \overline{\beta}_q) = e_q$ wurde schon im Beweis von (ii) aus (4.8) & (4.9) gefolgert. Daraus folgt

$$e_q = \mathrm{ggT}(\overline{\beta}_0, \overline{\beta}_1, \ldots, \overline{\beta}_q) \quad\implies\quad e_q \mid \overline{\beta}_i \quad \forall\, i \leq q$$

$e_{q-1} \nmid \overline{\beta}_q$, sonst würde folgen $e_{q-1} \mid \mathrm{ggT}(\overline{\beta}_0, \ldots, \overline{\beta}_q) = e_q$ ↯ zu $0 < e_q < e_{q-1}$

Sei $\gamma \in \Gamma$, $\gamma < \overline{\beta}_q$. Wegen $\overline{\beta}_0 < \ldots < \overline{\beta}_g$ folgt

$$\gamma \in \langle \overline{\beta}_0, \ldots, \overline{\beta}_{q-1} \rangle, \quad e_{q-1} \mid \overline{\beta}_i,\ 0 \le i \le q-1 \quad \Longrightarrow \quad e_{q-1} \mid \gamma$$

Also ist $\overline{\beta}_q$ das kleinste Element aus Γ, welches nicht durch e_{q-1} teilbar ist. Außerdem folgt aus $e_{q-1} \nmid \overline{\beta}_q$, daß $\overline{\beta}_q$ kein Element von $\langle \overline{\beta}_0, \ldots, \overline{\beta}_{q-1} \rangle$ ist. ▧

Korollar *Sei g das Geschlecht einer irreduziblen algebroiden Kurve R. Dann hat das minimale Erzeugendensystem der Wertehalbgruppe von R genauso viele Elemente wie R charakteristische Exponenten besitzt, nämlich $g+1$.*

Bemerkung: (4.9) ist eine rekursive Formel zur Berechnung der Wertehalbgruppe aus den charakteristischen Exponenten. Mit ihr haben wir für Körper der Charakteristik 0 explizit bewiesen, daß $\overline{\beta}_0, \ldots, \overline{\beta}_g$ allein von der Multiplizitätensequenz abhängen (siehe Satz 4.7 (ii) und Korollar zu Satz 2.6).

Wir kommen nun zur Berechnung des Grades des Konduktors von $\overline{R}$ in R. Wie schon bei der Definition bemerkt besteht unsere Aufgabe darin, das kleinste $c \in \mathbb{N}$ zu finden mit der Eigenschaft

$$c + \mathbb{N} \subset \Gamma$$

Sei $\mathfrak{K}$ der Konduktor von $\overline{R}$ in R. Es gilt nach dem *Theorem von Gorenstein-Rosenlicht* [Go]:

$$c = \dim_K(\overline{R}/\mathfrak{K}) = 2\dim_K(R/\mathfrak{K})$$

$$0 \to R/\mathfrak{K} \to \overline{R}/\mathfrak{K} \to \overline{R}/R \to 0$$

ist eine exakte Sequenz von (endlichdimensionalen) K-Vektorräumen, daher gilt

$$\delta = \dim_K(\overline{R}/R) = \dim_K(\overline{R}/\mathfrak{K}) - \dim_K(R/\mathfrak{K}) = \frac{c}{2} = \dim_K(R/\mathfrak{K})$$

Es gibt eine klassische Fomel für die δ-Invariante, nämlich

Lemma 4.10 *Sei $m_0, \ldots, m_n$ die Multiplizitätensequenz einer irreduziblen algebroiden Kurve R. Dann gilt für den Singularitätengrad von R:*

$$\delta = \sum_{i=0}^{n} \frac{m_i(m_i-1)}{2}$$

Beweis: Z. B. [Ha, V.3], example 3.9.3.

Daraus folgt sofort eine Formel für den Grad des Konduktors:

$$c = 2\delta = \sum_{i=0}^{n} m_i(m_i-1)$$

Da wir in der Lage sind, die m_i aus der Hamburger-Noether-Entwicklung zu berechnen, haben wir eine Formel für die Berechnung von c aus der HNE:

Seien $h_0, \ldots, h_{r-1}$ wie in der Generalvoraussetzung die Exponenten am Ende jeder Zeile der HNE von R, $n_i = \underline{\nu}(z_i)$, dann gilt

$$c = \sum_{i=0}^{r-1} h_i n_i (n_i - 1) \tag{4.10}$$

Der Beweis folgt direkt aus Lemma 4.10 und (4.1).

4.4 Die Schnittmultiplizität

Handelte es sich bisher immer um Invarianten einer irreduzible Kurve (oder eines Zweiges einer reduziblen Kurve), betrachten wir jetzt zwei Zweige einer reduziblen Kurve, deren Hamburger-Noether-Entwicklung wir kennen. Unser Ziel ist es, daraus ihre Schnittmultiplizität zu berechnen.

Folgende Eigenschaft der Schnittmultiplizität spielt für die Herleitung unserer Formel eine zentrale Rolle:

Lemma 4.11 *Seien $\mathcal{C}$ und $\mathcal{D}$ zwei irreduzible algebroide Kurven, seien $\mathcal{C}_1$ und $\mathcal{D}_1$ ihre quadratischen Transformierten und $e(\mathcal{C})$, $e(\mathcal{D})$ ihre Multiplizitäten. Dann gilt:*

$$\mathrm{mult}(\mathcal{C}, \mathcal{D}) = e(\mathcal{C})\, e(\mathcal{D}) + \mathrm{mult}(\mathcal{C}_1, \mathcal{D}_1)$$

Hierbei ist zu beachten, daß $\mathcal{C}$ und $\mathcal{D}$ als zwei Zweige einer reduziblen Kurve $\operatorname{Spec} O$ angesehen werden müssen, die einer gemeinsamen quadratischen Transformation unterworfen werden. $\mathcal{C}_1$ und $\mathcal{D}_1$ sind die korrespondierenden Zweige von $\operatorname{Spec} O_1$ (vgl. Definition 1.30). Falls sich $\mathcal{C}_1$ und $\mathcal{D}_1$ nicht mehr schneiden (d. h. O_1 ist kein lokaler Ring), definieren wir $\mathrm{mult}(\mathcal{C}_1, \mathcal{D}_1) := 0$

Beweis: Es bezeichnen R, S die lokalen Ringe: $\mathcal{C} =: \operatorname{Spec} R$, $\mathcal{D} =: \operatorname{Spec} S$. Vermöge einer gemeinsamen Einbettung beider Kurven in $\mathbb{A}^2 = \operatorname{Spec} k[[x, y]]$ erhalten wir zwei Isomorphismen

$$R \cong k[[x,y]] \,/\, \langle f \rangle\,, \quad S \cong k[[x,y]] \,/\, \langle g \rangle$$

so daß wir $\mathcal{C}$ und $\mathcal{D}$ mit den beiden Zweigen der Kurve

$$\operatorname{Spec} k[[x,y]] \,/\, \langle fg \rangle =: \operatorname{Spec} O$$

identifizieren können. O.B.d.A. seien die Isomorphismen so gewählt, daß

$$N := \underline{\nu}_R([x]_f) < \underline{\nu}_R([y]_f) =: M$$

gilt. Ferner seien $n := \underline{\nu}_S([x]_g)$, $m := \underline{\nu}_S([y]_g)$. Es gilt $e(\mathcal{C}) = N$.

Fall 1: $\mathcal{C}$ und $\mathcal{D}$ haben die gleiche Tangentialrichtung, d. h.

$$\Delta(\operatorname{Spec} O) = \{(1:0)\} \quad \text{und} \quad n < m$$

Dann ist $e(\mathcal{D}) = n$ und $O_1 = k[[x,y]] \,/\, \big\langle x^{-nN} f(x, xy)\, g(x, xy) \big\rangle$.

Sei $(x(t), y(t))$ eine primitive Parametrisierung von $\mathcal{C}$, dann gilt wegen $\operatorname{ord} x(t) = N < M = \operatorname{ord} y(t)$ nach Lemma 1.3:

$$\left(x(t), \frac{y(t)}{x(t)} \right) \quad \text{ist Parametrisierung von } \mathcal{C}_1$$

Außerdem ist $\mathcal{D}_1 = \operatorname{Spec} k[[x,y]] \,/\, \langle g_1 \rangle$ mit $g_1 = x^{-n} g(x, xy)$.

Daher gilt gemäß der Definition der Schnittmultiplizität

$$\begin{aligned}\operatorname{mult}(\mathcal{C}_1,\mathcal{D}_1) &= \operatorname{ord} g_1\left(x(t),\frac{y(t)}{x(t)}\right) = \operatorname{ord}\left(x(t)^{-n}\cdot g\left(x(t),\ x(t)\frac{y(t)}{x(t)}\right)\right)\\ &= -nN+\operatorname{ord} g(x(t),y(t)) = -e(\mathcal{C})\,e(\mathcal{D})+\operatorname{mult}(\mathcal{C},\mathcal{D})\end{aligned}$$

Fall 2: $\mathcal{C}$ und $\mathcal{D}$ schneiden sich transversal (d. h. $\operatorname{Spec} O$ hat mehrere Tangentialrichtungen).

Dann gilt $n \geq m$ und folglich $e(\mathcal{D}) = m$. Desweiteren schneiden sich $\mathcal{C}_1$ und $\mathcal{D}_1$ nicht mehr, wie in der Beobachtung auf S. 19 gesehen.

Unserer Vereinbarung gemäß ist demnach $\operatorname{mult}(\mathcal{C}_1,\mathcal{D}_1) = 0$ (wir könnten auch wie in Fall 1 verfahren und $\operatorname{ord} g_1(x(t),\, y(t)/x(t))$ berechnen – es kommt Null heraus, weil g_1 eine Einheit ist).

Wegen $e(\mathcal{D}) = m$ ist $g_m = c_0x^m + c_1x^{m-1}y + \ldots + c_my^m$ die homogene Leitform von g, und wegen Lemma 3.1 ist $c_0 \neq 0$. Weil $\operatorname{ord} x(t) = N < \operatorname{ord} y(t)$ ist, folgt

$$\operatorname{mult}(\mathcal{C},\mathcal{D}) = \operatorname{ord} g(x(t),y(t)) = \operatorname{ord} g_m(x(t),y(t)) = \operatorname{ord} x(t)^m = Nm = e(\mathcal{C})\,e(\mathcal{D})$$

▧

Ein weiteres Lemma benötigen wir noch zur Vorbereitung:

Lemma 4.12 *Seien $R = k[\![x,y]\!]\,/\,\langle f\rangle$ und $S = k[\![x,y]\!]\,/\,\langle g\rangle$ die lokalen Ringe zweier irreduzibler algebroider Kurven. Seien*

$$z_{-1} = a_{01}z_0 + \ldots \qquad \textit{bzw.} \qquad z'_{-1} = a'_{01}z'_0 + \ldots$$

die (nicht notwendigerweise von Null verschiedenen!) Anfangsterme von HNEs von R bzw. S in der Basis $\{[x],[y]\}$.

Falls $z_{-1} = [y]_f$ und $z'_{-1} = [y]_g$ oder $z_{-1} = [x]_f$ und $z'_{-1} = [x]_g$ ist, dann gilt für die quadratischen Transformierten der Kurven: Genau dann haben R und S einen gemeinsamen unendlich benachbarten Punkt in erster Umgebung (m.a.W.: R & S werden beim Aufblasen nicht getrennt), wenn $a_{01} = a'_{01}$ ist.

Sonst haben R und S genau dann einen gemeinsamen unendlich benachbarten Punkt, wenn $a_{01} \neq 0$ und $a'_{01} = 1/a_{01}$ ist.

Beweis: Durch Rückwärtseinsetzen erhalten wir Parametrisierungen von R und S, aus denen sich im ersten Fall als Tangentialrichtungen $(1 : a_{01})$ und $(1 : a'_{01})$ oder $(a_{01} : 1)$ und $(a'_{01} : 1)$ ablesen lassen (vgl. Definition). Im zweiten Fall lauten die Tangentialrichtungen $(1 : a_{01})$ und $(a'_{01} : 1)$ oder $(a_{01} : 1)$ und $(1 : a'_{01})$. Alles weitere ergibt sich aus Lemma 1.9 und Definition 1.30. ▧

Satz 4.13 (Zusammenhang zwischen Schnittmultiplizität und HNE) *Seien $R = k[\![x,y]\!]\,/\,\langle f\rangle$ und $S = k[\![x,y]\!]\,/\,\langle g\rangle$ die lokalen Ringe zweier irreduzibler algebroider Kurven mit Hamburger-Noether-Entwicklungen (in der Basis $\{[x],[y]\}$)*

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}z_j^i + z_j^{h_j}z_{j+1}, \quad 0 \leq j \leq r \qquad \textit{und}$$

$$z'_{j-1} = \sum_{i=1}^{h'_j} a'_{ji} {z'_j}^i + {z'_j}^{h'_j} z'_{j+1}, \quad 0 \leq j \leq r'$$

Sei $s \geq 0$ maximal mit der Eigenschaft

$$h_0 = h'_0, \; \ldots, \; h_{s-1} = h'_{s-1} \quad \textit{und} \quad a_{jk} = a'_{jk} \quad \forall\, j < s, \forall\, k \leq h_j$$

Wähle i minimal mit $a_{si} \neq a'_{si}$ *($i := h_s + 1$ oder $i := h'_s + 1$, falls alle existierenden Koeffizienten der HN-Matrizen in dieser Zeile übereinstimmen).*

Schließlich definieren wir für $n_j = \underline{\nu}_R(z_j)$ *und* $n'_j = \underline{\nu}_S(z'_j)$

$$S := \sum_{j=0}^{s-1} h_j\, n_j\, n'_j \qquad \textit{und} \qquad N := \sum_{j=0}^{s-1} h_j + i - 1$$

Falls die Hamburger-Noether-Entwicklungen so gewählt waren, daß $z_{-1} = [y]_f$ und $z'_{-1} = [y]_g$ oder $z_{-1} = [x]_f$ und $z'_{-1} = [x]_g$ ist, dann gilt:

(i) N ist die Anzahl der unendlich benachbarten Punkte, die beide Kurven R und S gemeinsam haben (das maximale Ideal O_0 von $k[\![x,y]\!] / \langle fg \rangle$ nicht mitgerechnet).

(ii) Für die Schnittmultiplizität von $\mathcal{C} := \operatorname{Spec} R$ und $\mathcal{D} := \operatorname{Spec} S$ gilt:

a) $\quad i \leq h_s, \quad i \leq h'_s \quad \Longrightarrow \quad \operatorname{mult}(\mathcal{C},\mathcal{D}) = S + i\, n_s\, n'_s$

b) $\quad i = h'_s + 1 \quad \Longrightarrow \quad \operatorname{mult}(\mathcal{C},\mathcal{D}) = S + h'_s\, n_s\, n'_s + n'_{s+1}\, n_s$

c) $\quad i = h_s + 1 \quad \Longrightarrow \quad \operatorname{mult}(\mathcal{C},\mathcal{D}) = S + h_s\, n'_s\, n_s + n_{s+1}\, n'_s$

Falls dagegen weder $[x]_f$ noch $[y]_g$ oder weder $[y]_f$ noch $[x]_g$ transversal sind (m.a.W.: falls wir die HNEs nicht wie im ersten Fall wählen können), dann haben $\mathcal{C}$ und $\mathcal{D}$ keinen unendlich benachbarten Punkt (außer O_0) gemeinsam und folglich ist

$$\operatorname{mult}(\mathcal{C},\mathcal{D}) = e(\mathcal{C})\, e(\mathcal{D}) = n_0\, n'_0$$

Beweis: Beginnen wir mit dem einfachen Teil: Falls $[x]_f$ und $[y]_g$ nicht transversal sind, sind beide HNEs eindeutig; es gilt

$$a_{01} = 0 = a'_{01}, \quad z_{-1} = [x]_f, \quad z'_{-1} = [y]_g$$

woraus mit Lemma 4.12 folgt, daß R und S keinen gemeinsamen unendlich benachbarten Punkt in erster Umgebung besitzen. Das bedeutet $\operatorname{mult}(\mathcal{C}_1, \mathcal{D}_1) = 0$, weswegen nach Lemma 4.11 $\operatorname{mult}(\mathcal{C},\mathcal{D}) = e(\mathcal{C})\, e(\mathcal{D})$ ist.
Der Fall „$[y]_f$ und $[x]_g$ sind nicht transversal“ geht analog.

Seien nun o.B.d.A. $[x]_f$ und $[x]_g$ transversal und beide HNEs so gewählt, daß $z_{-1} = [y]_f$ und $z'_{-1} = [y]_g$ ist. Wir bilden von beiden Ringen die Auflösungskette:

$$R =: R_0 \subseteq R_1 \subseteq \cdots \subseteq \overline{R}, \qquad S =: S_0 \subseteq S_1 \subseteq \cdots \subseteq \overline{S}$$

(i) Per def. von N und Satz 2.5 stimmen die Anfangsterme der HNEs von R_i und S_i überein für alle $0 \leq i < N$, während die Anfangsterme der HNEs von R_N und S_N verschieden sind. Daraus folgt mit Lemma 4.12, daß $R_0, \ldots, R_{N-1}$ und $S_0, \ldots, S_{N-1}$ je einen gemeinsamen unendlich benachbarten Punkt in erster Umgebung haben (insgesamt also N Punkte), während R_N und S_N beim Aufblasen voneinander getrennt werden.

(ii) beweisen wir mit Induktion über N:

I.A.: $h_0 + h_1 + \ldots + h_{s-1} + i - 1 = N = 0, \quad h_j > 0, \quad i > 0$
$\Rightarrow \ s = 0, \quad i = 1 \quad \Rightarrow \quad a_{01} \neq a'_{01}$
und die Voraussetzung von (a) gilt.

Wir haben also zu zeigen: $\operatorname{mult}(\mathcal{C}, \mathcal{D}) = S + n_0\, n'_0$. Das ist aber klar wegen $S = 0, \quad n_0 = e(\mathcal{C}), \quad n'_0 = e(\mathcal{D})$ und Lemmata 4.11 & 4.12.

I.V.: $N > 0$ und (ii) sei richtig für alle Kurven mit weniger als N gemeinsamen unendlich benachbarten Punkten.

Bemerkung: Die I.V. impliziert insbesondere, daß sich $\mathcal{C}_1, \mathcal{D}_1$ in einem gemeinsamen unendlich benachbarten Punkt in erster Umgebung von R schneiden, und daß (a) – (c) für $\mathcal{C}_1$ und $\mathcal{D}_1$ gilt (welche $N-1$ gemeinsame unendlich benachbarte Punkte besitzen). Nach Lemma 4.11 ist

$$\operatorname{mult}(\mathcal{C}, \mathcal{D}) = n_0\, n'_0 + \operatorname{mult}(\mathcal{C}_1, \mathcal{D}_1)$$

$$\text{Sei} \qquad \breve{z}_{j-1} = \sum_{i=1}^{\breve{h}_j} \breve{a}_{ji} \breve{z}_j^i + \breve{z}_j^{\breve{h}_j} \breve{z}_{j+1}, \qquad 0 \leq j \leq \breve{r} \qquad \text{die HNE von } R_1$$

$$\text{und} \qquad \breve{z}'_{j-1} = \sum_{i=1}^{\breve{h}'_j} \breve{a}'_{ji} \breve{z}'^{\,i}_j + \breve{z}'^{\,\breve{h}'_j}_j \breve{z}'_{j+1}, \qquad 0 \leq j \leq \breve{r}' \qquad \text{die HNE von } S_1$$

(beide in der Basis $\left\{[x], \frac{[y]}{[x]} - a_{01}\right\}$). Wir machen eine Fallunterscheidung:

Fall 1: $h_0 > 1, \quad h'_0 > 1$. Dann besteht zwischen den vier Hamburger-Noether-Entwicklungen folgender Zusammenhang (Satz 2.5):
$\breve{r} = r, \quad \breve{h}_j = h_j \ \forall\, 0 < j \leq r, \quad \breve{h}_0 = h_0 - 1, \quad \breve{n}_j := \underline{\nu}_{R_1}(\breve{z}_j) = n_j \ \forall\, j$
und das gleiche für $\breve{r}'$, $\breve{h}'_j$ und $\breve{n}'_j$.
Wir definieren $\breve{s}$, $\breve{\imath}$ und $\breve{S}$ analog zu s, i und S, dann folgt sofort $\breve{s} = s$.

Fall 1.1: $s = 0 \quad \Longrightarrow \quad \breve{\imath} = i - 1, \quad \breve{S} = 0 = S$

1.1 (a) $i \leq h_0, \quad i \leq h'_0 \iff \breve{\imath} \leq \breve{h}_0, \quad \breve{\imath} \leq \breve{h}'_0$

$$\Longrightarrow \quad \operatorname{mult}(\mathcal{C}, \mathcal{D}) = n_0\, n'_0 + (i-1) n_0\, n'_0 = S + i\, n_0\, n'_0 \quad \checkmark$$

1.1 (b) $i = h'_0 + 1 \iff \breve{\imath} = \breve{h}'_0 + 1$

$$\Longrightarrow \quad \operatorname{mult}(\mathcal{C}, \mathcal{D}) = n_0\, n'_0 + (h'_0 - 1) n_0\, n'_0 + n'_1\, n_0 = S + h'_0\, n_0\, n'_0 + n'_1\, n_0 \quad \checkmark$$

1.1 (c) $i = h_0 + 1 \iff \breve{\imath} = \breve{h}_0 + 1$: analog zu 1.1 (b)

Fall 1.2: $s > 0 \quad \Longrightarrow \quad \breve{\imath} = i$ und es gilt

$$\breve{S} = \sum_{j=0}^{s-1} \breve{h}_j \, n_j \, n'_j = (h_0 - 1) n_0 \, n'_0 + \sum_{j=1}^{s-1} h_j \, n_j \, n'_j = S - n_0 \, n'_0$$

$$\Longrightarrow \qquad \operatorname{mult}(\mathcal{C}, \mathcal{D}) - S = \operatorname{mult}(\mathcal{C}_1, \mathcal{D}_1) - \breve{S}$$

Weil $\breve{h}_{\breve{s}} = h_s$, $\breve{h}'_{\breve{s}} = h'_s$, $\breve{\imath} = i$ und $\forall j : \breve{n}_j = n_j$, $\breve{n}'_j = n'_j$ gilt, ist die Formel für $\operatorname{mult}(\mathcal{C}, \mathcal{D})$ in allen Fällen (a) – (c) richtig.

Fall 2: $h_0 = 1 = h'_0$. Dann gilt (Satz 2.5):
$\breve{r} = r - 1, \quad \forall\, 0 \le j < r : \quad \breve{h}_j = h_{j+1}, \quad \breve{n}_j = n_{j+1}$
und das gleiche für $\breve{r}'$, $\breve{h}'_j$ und $\breve{n}'_j$. Daraus ergibt sich $\breve{s} = s - 1$ und $\breve{\imath} = i$.

$$\breve{S} = \sum_{j=0}^{s-2} h_{j+1} \, n_{j+1} \, n'_{j+1} = S - h_0 \, n_0 \, n'_0 = S - n_0 \, n'_0$$

Wie im Fall 1.2 gilt also $\operatorname{mult}(\mathcal{C}, \mathcal{D}) - S = \operatorname{mult}(\mathcal{C}_1, \mathcal{D}_1) - \breve{S}$ und wieder $\breve{h}_{\breve{s}} = h_s$, $\breve{h}'_{\breve{s}} = h'_s$. Daraus folgt die Gültigkeit von (a), (b) und (c).

Fall 3: O.B.d.A. $h_0 = 1$, $h'_0 > 1$ (sonst vertausche R und S).
Dann gilt: $s = 0$, $\breve{r} = r - 1$, $\breve{r}' = r'$, und $\breve{h}_j$, $\breve{n}_j$ sind wie in Fall 2, $\breve{h}'_j$, $\breve{n}'_j$ wie in Fall 1.
Wegen $N > 0$ (I.V.) ist $a_{01} = a'_{01} \Rightarrow i > 1$. Da aber $h_0 = 1$ ist, muß $i = 2$ sein.
Nach Satz 2.5 ist $\breve{z}_{-1} = z_0 = [x]_f$, $\breve{z}_0 = z_1 = \frac{[y]_f}{[x]_f} - a_{01}$, jedoch $\breve{z}'_{-1} = \frac{[y]_g}{[x]_g} - a_{01}$, $\breve{z}'_0 = z'_0 = [x]_g$ (d. h. $[x]_g$ ist in S_1 weiter transversal, aber $[x]_f$ ist es in R_1 *nicht* mehr).
Wegen $\breve{a}_{01} = a_{11} = 0$ folgt mit Lemma 4.12, daß R_1 und S_1 keinen gemeinsamen unendlich benachbarten Punkt in erster Umgebung besitzen, d. h.:

$$\operatorname{mult}(\mathcal{C}_2, \mathcal{D}_2) = 0 \quad \underset{4.11}{\Longrightarrow} \quad \operatorname{mult}(\mathcal{C}_1, \mathcal{D}_1) = e(\mathcal{C}_1) \, e(\mathcal{D}_1) = \breve{n}_0 \, \breve{n}'_0$$

Daraus erhalten wir

$$\operatorname{mult}(\mathcal{C}, \mathcal{D}) = n_0 \, n'_0 + \breve{n}_0 \, \breve{n}'_0 = S + h_0 \, n_0 \, n'_0 + n_1 \, n'_0 \quad \checkmark$$

denn wegen $s = 0$ ist $S = 0$, und wir befinden uns im Fall (c) $(i = h_0 + 1)$.

▧

Bemerkung: Mit der Kenntnis der Anzahl der gemeinsamen unendlich benachbarten Punkte zweier Kurven (vgl. 4.13 (i)) können wir auch die Berechnung der Multiplizitätensequenz einer reduziblen Kurve vervollständigen: Sei R eine reduzible Kurve; für jeden Zweig von R sei die HNE bekannt.

- Eine geeignete Sortierung der Zweige $R^1, \dots, R^r$ von R ist dadurch charakterisiert, daß gilt:

  $\forall\, k > 0,\ \forall\, 1 \le i \le r$: falls sich die k-ten Transformierten R_k^i und R_k^{i+1} nicht schneiden, dann schneiden sich auch R_k^i und R_k^j nicht $\forall\, j > i$

  Aus Satz 4.13 geht hervor, daß die Reihenfolge der Hamburger-Noether-Entwicklungen $\mathsf{HN}_1, \dots, \mathsf{HN}_r$, die man erhält, wenn man die (von `HNE-reduzibel` berechneten) Auflösungsbäume in natürlicher[1] Reihenfolge durchläuft, diese Eigenschaft besitzt. Hier ist also nichts mehr zu tun.

- Für alle $1 \le i \le r$ berechne die Multiplizitätensequenz $m_0^i, m_1^i, \dots$ von R^i aus HN_i (mit der Formel 4.1).

- Zur richtigen Gruppierung der Multiplizitäten vergleiche die gemeinsamen unendlich benachbarten Punkte zweier (in der Sortierung) aufeinanderfolgender Zweige:

  $$\forall\, k \ge 0,\ \forall\, 1 \le i < r : \quad m_k^{i+1} \text{ gehört zur gleichen Gruppe wie } m_k^i \iff$$
  $$\text{die Anzahl gemeinsamer unendl. benachb. Pkte. von } R^i, R^{i+1} \text{ ist } \ge k$$

- Wir können abbrechen, wenn k so groß ist, daß $m_k^i = 1\ \forall\, i$ gilt und $k >$ Anzahl der gemeinsamen unendlich benachbarten Punkte zweier beliebiger Zweige von R ist (die Sequenz wird konstant).

Beispiel: Sei $f := x(y^2 + x^3y + x^3)(y^2 + x^5y + x^3)(y^5 + x^2) \in \overline{\mathbb{F}_2}[[x, y]]$
Die durch f gegebene Kurve hat vier Zweige mit folgenden Hamburger-Noether-Entwicklungen (vgl. Beispiel 3 auf S. 92):

$$\mathsf{HN}_1 : \begin{array}{lcl} [y] = z_{-1} & = & z_0 z_1 \\ [x] = \ \ z_0 & = & z_1^2 + z_1^5 + \dots \end{array} \qquad \mathsf{HN}_2 : \begin{array}{lcl} [y] = z_{-1} & = & z_0 z_1 \\ [x] = \ \ z_0 & = & z_1^2 + z_1^9 + \dots \end{array}$$

$$\mathsf{HN}_3 : \begin{array}{lcl} [x] = z_{-1} & = & 0 \\ [y] = \ \ z_0 & & \end{array} \qquad \mathsf{HN}_4 : \begin{array}{lcl} [x] = z_{-1} & = & z_0^2 z_1 \\ [y] = \ \ z_0 & = & z_1^2 \end{array}$$

Diese sind bereits geeignet sortiert. Mit Anwendung von (4.1) erhalten wir als Multiplizitätensequenz von

$$\mathsf{HN}_1 : 2, 1, 1, \dots, \quad \mathsf{HN}_2 : 2, 1, 1, \dots, \quad \mathsf{HN}_3 : 1, 1, \dots, \quad \mathsf{HN}_4 : 2, 2, 1, 1, \dots$$

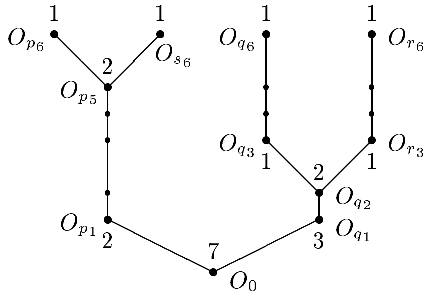
Sei $N(i, j)$ die Anzahl der gemeinsamen unendlich benachbarten Punkte von i-tem und j-tem Zweig (die Zweige mit $\mathsf{HN}_i, \mathsf{HN}_j$ als HNE). Für $N(1, 2)$ und $N(3, 4)$ tritt Satz 4.13 (i) in Kraft, für $N(2, 3)$ gilt die Alternative von 4.13, d. h.

$$N(1, 2) = 5, \quad N(2, 3) = 0, \quad N(3, 4) = 2$$

[1] in dem Sinn, daß jeder Teilbaum vollständig ausgewertet wird, bevor HNEs, die mit Blättern aus anderen Teilbäumen korrespondieren, berechnet werden. Also T ein Teilbaum, $i < j$, $\mathsf{HN}_i, \mathsf{HN}_j$ gehören zu Blättern aus $T \Rightarrow \mathsf{HN}_k$ gehört zu einem Blatt von T für alle $i \le k \le j$

Damit lautet die Multiplizitätensequenz von $\overline{\mathbb{F}_2}[\![x,y]\!] \,/\, \langle f \rangle$:

	Anzahl quadr. Transf.
$[(2,2,1,2)]$	0
$[(1,1),(1,2)]$	1
$[(1,1),(1,1)]$	2
$[(1,1),(1),(1)]$	3
$[(1,1),(1),(1)]$	4
$[(1,1),(1),(1)]$	5
$[(1),(1),(1),(1)]$	6
....	≥ 7



Rechts im Bild befindet sich die Auflösungskette der Kurve, die wir direkt an der Multiplizitätensequenz ablesen können. Die Zahlen bei den unendlich benachbarten Punkten sind die Multiplizitäten der durch diese Punkte verlaufenden Kurven. $N(i,j)$ sind die Ebenen, wo der Baum verzweigt.

Wir berechnen ein paar Schnittmultiplizitäten, dazu sei $\mathcal{C}_i$ der Zweig mit HN_i als HNE. Um die Probe zu machen identifizieren wir die irreduziblen Faktoren von f, die zu $\mathcal{C}_i$ gehören: Offensichtlich ist $(0,t)$ eine Parametrisierung von $\mathcal{C}_3$ und (t^5,t^2) eine von $\mathcal{C}_4$ (rückwärtseinsetzen). Also gehört x zu $\mathcal{C}_3$ und y^5+x^2 zu $\mathcal{C}_4$. Um zu entscheiden, welcher der beiden verbleibenden Faktoren zu $\mathcal{C}_1$ gehört, kann man z. B. die HNE von $y^2+x^3y+x^3$ berechnen. Da wir in Beispiel 3 aber alle Transformationen an den Faktoren von f getrennt durchgeführt haben, haben wir dort schon gesehen, daß HN_1 von $y^2+x^3y+x^3$ und HN_2 von $y^2+x^5y+x^3$ herrührt.

$\mathbf{mult}(\boldsymbol{\mathcal{C}_1},\boldsymbol{\mathcal{C}_2})$: Mit Bezeichnungen aus Satz 4.13 gilt:

$$s=1, \quad i=5 \quad \text{erfüllt die Voraussetzung von (ii) (a)}, \quad S=4$$

$$\Longrightarrow \quad \operatorname{mult}(\mathcal{C}_1,\mathcal{C}_2) = 4+5\cdot 1 = 9$$

Probe: Rückwärtseinsetzen liefert $(x(t),y(t)) = \big(t^2+t^5+O(t^6), t^3+t^6+O(t^7)\big)$ als Parametrisierung von $\mathcal{C}_1$, weswegen per definitionem

$$\operatorname{mult}(\mathcal{C}_1,\mathcal{C}_2) = \operatorname{ord}_t\big(y(t)^2+x(t)^5y(t)+x(t)^3\big) = \operatorname{ord}_t\big(t^9+O\big(t^{10}\big)\big) = 9 \text{ ist.}$$

$\mathbf{mult}(\boldsymbol{\mathcal{C}_1},\boldsymbol{\mathcal{C}_3})$: Nach Satz 4.13 gilt:

$$\operatorname{mult}(\mathcal{C}_1,\mathcal{C}_3) = e(\mathcal{C}_1)\,e(\mathcal{C}_2) = 2\cdot 1 = 2$$

Probe: $\quad \operatorname{mult}(\mathcal{C}_1,\mathcal{C}_3) = \operatorname{ord}_t x(t) = \operatorname{ord}_t\big(t^2+t^5+O\big(t^6\big)\big) = 2.$

$\mathbf{mult}(\boldsymbol{\mathcal{C}_3},\boldsymbol{\mathcal{C}_4})$: Hier gilt:

$$s=0, \quad i=3 \quad \text{erfüllt die Voraussetzung von (ii) (b)}, \quad S=0$$

$$\Longrightarrow \quad \operatorname{mult}(\mathcal{C}_3,\mathcal{C}_4) = 0+2\cdot 1\cdot 2+1 = 5$$

Probe: $\quad \operatorname{mult}(\mathcal{C}_3,\mathcal{C}_4) = \operatorname{ord}_t\big(t^5\big) = 5.$

4.5 Abgeleitete Umrechnungsformeln

Bisher haben wir Formeln entwickelt, um Kurveninvarianten direkt oder indirekt aus der Hamburger-Noether-Entwicklung berechnen zu können. Wir ergänzen einige weitere Formeln, um verschiedene Invarianten untereinander umrechnen zu können. Sie beruhen im wesentlichen auf den bereits bewiesenen Formeln. Als Nebenprodukt erhalten wir Aussagen, welche Invarianten noch ein vollständiges System bilden.

Wir setzen generell voraus, daß R der lokale Ring einer irreduziblen algebroiden Kurve ist.

Satz 4.14 *Die charakteristischen Exponenten einer irreduziblen algebroiden Kurve sind ein vollständiges System von Invarianten für die Gleichauflösung.*

Um es konkreter zu machen: Seien $\beta_0, \ldots, \beta_g$ die charakteristischen Exponenten von R. Falls $g = 0$ ist, so ist $1, 1, \ldots$ die Multiplizitätensequenz von R.

Falls $g \geq 1$ ist, definieren wir $h_i, n_i \in \mathbb{N}$ durch den euklidischen Algorithmus: Es gelte $n_0 := \beta_0$ und

$$\begin{array}{rcll} \beta_1 & = & h_0 n_0 + n_1, & \qquad 0 < n_1 < n_0 \\ n_0 & = & h_1 n_1 + n_2, & \qquad 0 < n_2 < n_1 \\ & & \cdots\cdots\cdots & \\ n_l & = & h_{l+1} n_{l+1} & \end{array}$$

Dann ist $l \geq 0$, $h_i > 0$ für alle i und die Multiplizitätensequenz von R beginnt wie folgt:

$$\underbrace{n_0, \ldots, n_0}_{h_0}, \underbrace{n_1, \ldots, n_1}_{h_1}, n_2, \ldots\ldots, n_{l-1}, \underbrace{n_l, \ldots, n_l}_{h_l}, n_{l+1}, \ldots\ldots$$

Beweis: Wir beginnen mit dem zweiten Teil: Falls $g = 0$ ist, ist die Multiplizität von R bereits 1 und die Multiplizitätensequenz folglich trivial.

Sei nun $g > 0$. Per definitionem gilt $n_0 = \beta_0 < \beta_1$ und $\beta_0 \nmid \beta_1$, woraus $h_0 > 0$ und $l \geq 0$ folgt. $h_i > 0$ für $i > 0$ folgt aus der Wahl des Restes $n_i < n_{i-1}$.

Es bezeichne wie gewohnt R_i die i-te quadratische Transformierte von R. Wegen $e(R) = n_0$ und Lemma 4.6 (a) gilt: $\underbrace{n_0, n_0, \ldots, n_0}_{h_0}, \ldots$ ist der Anfang der Multiplizitätensequenz von R, und $n_0, n_0 + n_1$ sind die ersten beiden charakteristischen Exponenten von R_{h_0-1}. Falls $l > 0$ ist, sind nach 4.6 (b) n_1 und n_0 die ersten beiden charakteristischen Exponenten von R_{h_0}, d. h. wir können die gleiche Argumentation für R_{h_0} statt für R wiederholen – die Multiplizitätensequenz von R_{h_0} beginnt mit $\underbrace{n_1, n_1, \ldots, n_1}_{h_1}, \ldots$, die von R folglich mit $\underbrace{n_0, \ldots, n_0}_{h_0}, \underbrace{n_1, \ldots, n_1}_{h_1}, \ldots$

Mit Induktion erhalten wir $\underbrace{n_0, \ldots, n_0}_{h_0}, \ldots, \underbrace{n_l, \ldots, n_l}_{h_l}, \ldots$ als Anfang der Multiplizitätensequenz von R und $n_l, n_l + n_{l+1}$ als erste zwei charakteristische Exponenten von $R_{h_0+\ldots+h_l-1}$. Weil n_l von n_{l+1} geteilt wird, tritt

Fall (c) von 4.6 ein, woraus wir (immerhin noch) $e(R_{h_0+\ldots+h_l}) = n_{l+1}$ folgern können. Das ist die letzte Multiplizität in der behaupteten Sequenz.

Nun zum ersten Teil der Behauptung: Durch die Formel in Satz 4.7 sind die charakteristischen Exponenten eindeutig durch die Multiplizitätensequenz bestimmt. Sind umgekehrt die charakteristischen Exponenten β_i von R bekannt, dann können wir mit dem bereits Bewiesenen die ersten $h_0 + \ldots + h_l + 1$ Multiplizitäten in der Auflösungskette explizit angeben; das sind mindestens die ersten beiden. Aus Lemma 4.6 folgt, daß durch die β_i bereits die charakteristischen Exponenten aller Glieder der Auflösungskette festgelegt sind (z. B. auch die von $R_{h_0+\ldots+h_l}$). Also ist die ganze Multiplizitätensequenz von R eindeutig durch die β_i bestimmt. ▧

Korollar *Die Puiseux-Paare einer irreduziblen algebroiden Kurve sind ein vollständiges System von Invarianten für die Gleichauflösung.*
Beweis: Satz 4.14 und Lemma 4.4 ▧

Lemma 4.15 (Formeln für die Halbgruppe) *Seien $\overline{\beta}_0, \ldots, \overline{\beta}_g$ die minimalen Erzeuger der Wertehalbgruppe von R, $\beta_0, \ldots, \beta_g$ die charakteristischen Exponenten und $(m_1, n_1), \ldots, (m_g, n_g)$ die Puiseux-Paare von R. Dann gilt für alle $2 \le i \le g$:*

(i)
$$\overline{\beta}_0 = \beta_0, \quad \overline{\beta}_1 = \beta_1, \quad \overline{\beta}_i = n_{i-1}\overline{\beta}_{i-1} - \beta_{i-1} + \beta_i \tag{4.9}$$

(ii)
$$\begin{aligned} &\overline{\beta}_0 = n_1 n_2 \cdots n_g, \quad \overline{\beta}_1 = m_1 n_2 \cdots n_g, \\ &\overline{\beta}_i = n_{i-1}\overline{\beta}_{i-1} + (m_i - m_{i-1}n_i) n_{i+1} \cdots n_g \end{aligned} \tag{4.11}$$

(iii)
$$\beta_0 = \overline{\beta}_0, \quad \beta_1 = \overline{\beta}_1, \quad \beta_i = \overline{\beta}_i - n_{i-1}\overline{\beta}_{i-1} + \beta_{i-1} \tag{4.12}$$

(iv) *Mit* $d_i := \left(\overline{\beta}_i - n_{i-1}\overline{\beta}_{i-1}\right) n_1 \cdots n_{i-1}$ *gilt*

$$\begin{aligned} n_1 &= \frac{\overline{\beta}_0}{\mathrm{ggT}(\overline{\beta}_0, \overline{\beta}_1)}, \qquad & n_i &= \frac{\overline{\beta}_0}{\mathrm{ggT}(\overline{\beta}_0, d_i)} \\ m_1 &= \frac{\overline{\beta}_1}{\mathrm{ggT}(\overline{\beta}_0, \overline{\beta}_1)}, \qquad & m_i &= \frac{d_i}{\mathrm{ggT}(\overline{\beta}_0, d_i)} + m_{i-1} n_i \end{aligned} \tag{4.13}$$

Beweis: (i) ist eine Wiederholung der Formel auf Seite 123,
(ii) folgt aus (i) und Lemma 4.4 und
(iii) folgt ebenfalls aus (i).

Zu (iv): Die Formel für m_1 und n_1 folgt aus (i) und Lemma 4.4.
Für $i \ge 2$ folgt aus (ii)

$$\begin{aligned} &m_i n_{i+1} \cdots n_g = \overline{\beta}_i - n_{i-1}\overline{\beta}_{i-1} + m_{i-1} n_i \cdots n_g \\ &\qquad \Longrightarrow \quad \frac{m_i}{n_i} = \frac{\overline{\beta}_i - n_{i-1}\overline{\beta}_{i-1}}{n_i \cdots n_g} + m_{i-1} = \frac{d_i}{\overline{\beta}_0} + m_{i-1} \\ &\mathrm{ggT}(m_i - m_{i-1}n_i, n_i) = \mathrm{ggT}(m_i, n_i) = 1 \\ &\qquad \Longrightarrow \quad m_i - m_{i-1}n_i = \frac{d_i}{\mathrm{ggT}\left(\overline{\beta}_0, d_i\right)}, \quad n_i = \frac{\overline{\beta}_0}{\mathrm{ggT}\left(\overline{\beta}_0, d_i\right)} \end{aligned}$$

▧

Korollar *Das minimale Erzeugendensystem der Wertehalbgruppe ist ein vollständiges System von Invarianten für die Gleichauflösung.*
Beweis: (4.11), (4.13) und Korollar zu 4.14 ▧

Zum Schluß entwickeln wir noch eine Formel, mit der man den Grad des Konduktors (und damit natürlich auch die δ-Invariante $\delta = c/2$) aus den charakteristischen Exponenten berechnen kann.

Lemma 4.16 *Seien $\beta_0, \ldots, \beta_g$ die charakteristischen Exponenten von R, c der Grad des Konduktors von $\overline{R}$ in R und $e_i := \mathrm{ggT}(\beta_0, \ldots, \beta_i)$, $0 \leq i \leq g$. Dann ist*

$$c = \beta_g(e_{g-1} - 1) + \beta_{g-1}(e_{g-2} - e_{g-1}) + \ldots + \beta_1(\beta_0 - e_1) - (\beta_0 - 1) \qquad (4.14)$$

Wenn $s_0, \ldots, s_g$ die Daten aus der HNE von R in reduzierter Form sind (wie auf Seite 114 definiert) und $n_j = \underline{\nu}(z_j)$, dann gilt

$$\forall\, 0 \leq i \leq g: \quad e_i = n_{s_i}$$

weswegen wir die Formel für c auch folgendermaßen schreiben können:

$$c = \beta_g\left(n_{s_{g-1}} - 1\right) + \beta_{g-1}\left(n_{s_{g-2}} - n_{s_{g-1}}\right) + \ldots + \beta_1\left(n_0 - n_{s_1}\right) - \left(n_0 - 1\right) \quad (4.15)$$

Beweis: Sei R_1 die erste quadratische Transformierte von R, c_1 der Grad des Konduktors von $\overline{R}$ in R_1 und $n := e(R)$ die Multiplizität, dann gilt nach Lemma 4.10

$$c = c_1 + n(n-1) \qquad (*)$$

Wir zeigen die erste Formel mit vollständiger Induktion:
Falls $e(R) = 1$ ist, ist die Halbgruppe $\Gamma = \mathbb{N} \implies c = 0$ (oder man verwendet Lemma 4.10, welches den selben Wert liefert).
Desweiteren ist $g = 0$ und $\beta_0 = e(R) = 1$, woraus sich ergibt:

$$-(\beta_0 - 1) = 0 = c \quad \checkmark$$

I.V.: Seien $\beta'_0, \ldots, \beta'_{g'}$ die charakteristischen Exponenten von R_1, $e'_i := \mathrm{ggT}(\beta'_0, \ldots, \beta'_i)$ und es gelte

$$c_1 = \beta'_{g'}(e'_{g'-1} - 1) + \beta'_{g'-1}(e'_{g'-2} - e'_{g'-1}) + \ldots + \beta'_1(\beta'_0 - e'_1) - (\beta'_0 - 1)$$

I.S.: Es gilt einer der drei Fälle von Lemma 4.6:

(a) $g = g', \quad \beta_0 = \beta'_0, \quad \beta'_\nu = \beta_\nu - n \quad \forall\, 1 \leq \nu \leq g \quad \implies \quad e'_i = e_i \quad \forall\, i$

$$\begin{aligned} \implies \quad c_1 \;=\; & \beta_g(e_{g-1} - 1) + \beta_{g-1}(e_{g-2} - e_{g-1}) + \ldots + \beta_1(\beta_0 - e_1) \\ & - n\underbrace{(e_{g-1} - 1 + e_{g-2} - e_{g-1} + \ldots + n - e_1)}_{n-1} - (\beta_0 - 1) \end{aligned}$$

$$\underset{(*)}{\implies} \quad c = \beta_g(e_{g-1} - 1) + \beta_{g-1}(e_{g-2} - e_{g-1}) + \ldots + \beta_1(\beta_0 - e_1) - (\beta_0 - 1)$$

(b) $\quad g = g', \quad \beta'_0 = \beta_1 - n, \quad \beta'_\nu = \beta_\nu - \beta_1 + n \quad \forall\, 1 \leq \nu \leq g$

$$\begin{aligned}
\Longrightarrow \quad e'_i &= e_i \quad \forall\, i \geq 1 \\
\Longrightarrow \quad c_1 &= \beta_g(e_{g-1} - 1) + \beta_{g-1}(e_{g-2} - e_{g-1}) + \ldots + \beta_2(e_1 - e_2) \\
&\quad + n(\beta_1 - n - e_1) - (\beta_1 - n - 1) \\
&\quad + (n - \beta_1)(e_{g-1} - 1 + e_{g-2} - e_{g-1} + \ldots + e_1 - e_2) \\
&= \beta_g(e_{g-1} - 1) + \ldots + \beta_2(e_1 - e_2) + (\beta_1 - n)(n-1) - ne_1 + 1 \\
&\quad + (n - \beta_1)(e_1 - 1) \\
&= \beta_g(e_{g-1} - 1) + \ldots + \beta_2(e_1 - e_2) + (\beta_1 - n)(n - e_1) - ne_1 + 1 \\
&= \beta_g(e_{g-1} - 1) + \ldots + \beta_2(e_1 - e_2) + \beta_1(n - e_1) - n^2 + 1 \\
\underset{(*)}{\Longrightarrow} \quad c &= \beta_g(e_{g-1} - 1) + \ldots + \beta_1(\beta_0 - e_1) - (\beta_0 - 1)
\end{aligned}$$

(c) $\quad g = g'+1, \quad \beta'_0 = \beta_1 - n, \quad \beta'_\nu = \beta_{\nu+1} - \beta_1 + n \quad \forall 1 \leq \nu \leq g-1, \quad \beta_1 - n \mid n$

$$\begin{aligned}
\Longrightarrow \quad e'_1 &= \mathrm{ggT}(\beta_1 - n, \beta_2 - \beta_1 + n) = \mathrm{ggT}(\beta_1 - n, \beta_2) \mid n \\
\Longrightarrow \quad e'_1 &= \mathrm{ggT}(n, e'_1) = \mathrm{ggT}(n, \beta_1, \beta_2) = e_2 \\
\Longrightarrow \quad e'_i &= \mathrm{ggT}(e'_{i-1}, \beta'_i) = \mathrm{ggT}(e_i, \beta_i) = e_{i+1} \quad \forall\, 2 \leq i \leq g-1 \\
\Longrightarrow \quad c_1 &= \beta_g(e_{g-1} - 1) + \beta_{g-1}(e_{g-2} - e_{g-1}) + \ldots + \beta_3(e_2 - e_3) \\
&\quad + \beta_2(\beta_1 - n - e_2) - (\underline{\beta_1 - n - 1}) + (n - \beta_1) \cdots \\
&\quad \cdots (e_{g-1} - 1 + e_{g-2} - e_{g-1} + \ldots + e_2 - e_3 + \beta_1 - n - e_2) \\
&= \beta_g(e_{g-1} - 1) + \ldots + \beta_3(e_2 - e_3) - \beta_2 e_2 \\
&\quad + \beta_2(\beta_1 - n) + (n - \beta_1 - \underline{1})(\beta_1 - n - 1) \\
e_1 &= \mathrm{ggT}(n, \beta_1) = \mathrm{ggT}(n, \beta_1 - n) \underbrace{=}_{\beta_1 - n \mid n} \beta_1 - n \\
\Longrightarrow \quad c_1 &= \beta_g(e_{g-1} - 1) + \ldots + \beta_3(e_2 - e_3) - \beta_2 e_2 \\
&\quad + \beta_2 e_1 + (n-1)(\beta_1 - n - 1) - \beta_1(e_1 - 1) \\
&= \beta_g(e_{g-1} - 1) + \ldots + \beta_3(e_2 - e_3) + \beta_2(e_1 - e_2) + \beta_1(n - e_1) \\
&\quad - n(n-1) - n + 1 \\
\underset{(*)}{\Longrightarrow} \quad c &= \beta_g(e_{g-1} - 1) + \ldots + \beta_2(e_1 - e_2) + \beta_1(\beta_0 - e_1) - (\beta_0 - 1)
\end{aligned}$$

In allen drei Fällen kommt also die gleiche Formel für den Grad des Konduktors heraus, und diese wollten wir auch zeigen.

Zur Behauptung $\;\mathrm{ggT}(\beta_0, \ldots, \beta_i) = n_{s_i}\;$ für alle i: Aus Lemma 4.1 folgt

$$\left.\begin{array}{lcl}
j \notin \{s_0, \ldots, s_g\} & \Longrightarrow & n_{j-1} = h_j n_j + n_{j+1} \\
j = s_\nu \text{ für ein } \nu & \Longrightarrow & \exists\, l_\nu : n_{j-1} = l_\nu n_j
\end{array}\right\} (\aleph)$$

Sei $\;0 \leq \nu < g$. Aus $(\aleph)$ folgt, daß $n_{s_{\nu+1}}$ der letzte Rest ist, der im euklidischen Algorithmus angewandt auf n_{s_ν} und $n_{s_\nu+1}$ auftritt. Daher ist

$$n_{s_{\nu+1}} = \mathrm{ggT}(n_{s_\nu}, n_{s_\nu+1}) \tag{4.16}$$

und $n_{s_{\nu+1}}$ ist ein gemeinsamer Teiler von $n_{s_\nu}, n_{s_\nu+1}, \ldots, n_{s_{\nu+1}-1}, n_{s_{\nu+1}}$. $(\star)$
Wir verwenden wieder Induktion:

I.A.: $s_0 = 0 \quad \Longrightarrow \quad n_{s_0} = n_0 = \beta_0 = e_0$

I.V.: $\mathrm{ggT}(\beta_0, \ldots, \beta_{\nu+1}) = \mathrm{ggT}(n_{s_\nu}, \beta_{\nu+1})$

I.S.: Aus $(\star)$ folgt $n_{s_\nu} \,|\, n_j \quad \forall\, 0 \leq j \leq s_\nu$. Daher ist mit Satz 4.7

$$e_{\nu+1} = \mathrm{ggT}(n_{s_\nu}, \beta_{\nu+1}) \underset{(4.5)}{=} \mathrm{ggT}(n_{s_\nu}, n_{s_\nu+1}) \underset{(4.16)}{=} n_{s_{\nu+1}}$$

Die Formel (4.15) folgt nun trivial aus der vorhergehenden Formel für c. ▧

Teil II

Implementierung

Die zweite Hälfte dieser Arbeit beschäftigt sich mit der Implementation des Algorithmus zur Berechnung der Hamburger-Noether-Entwicklung in Singular, sowie der Gewinnung weiterer Informationen aus der HNE. Dazu stellen wir zunächst zusammen, auf welche von Singular bereitgestellte Funktionen diese Implementation zurückgreifen kann. Daneben werden Funktionen benötigt, die in Singular noch nicht vorhanden waren, und deren Programmierung ein Teil der vorliegenden Arbeit wurde. Die wichtigsten sind ein Quadratfreiheitstest für Polynome (5.1), die Berechnung des Newtonpolygons (5.2.1) und der Wechsel in algebraische Körpererweiterungen (wozu ein primitives Element berechnet werden muß, 5.4).

Diese und die Prozeduren zur Berechnung der Hamburger-Noether-Entwicklung und der Kurveninvarianten werden neben weiteren wichtigen Routinen in Kapitel 5 vorgestellt. Abschließend untersuchen wir in Kapitel 6 die Performance der beiden Prozeduren, die die HNE im irreduziblen bzw. im reduziblen Fall berechnen (die Berechnung der Invarianten braucht praktisch keine weiteren Ressourcen). Dazu stellen wir einige Beispiele vor, mit denen die Prozeduren getestet wurden, und untersuchen ihren Zeit- und Speicherbedarf. Im Laufe der Entwicklung der Bibliothek wurden einige Verbesserungen an der Implementierung vorgenommen, deren Auswirkungen wir in Abschnitt 6.2 betrachten werden.

Voraussetzungen des Systems

Folgende Funktionen von Singular werden zur Realisierung der Algorithmen benötigt (der Name der Funktion erscheint in Schreibmaschinenschrift; zur näheren Erläuterung sei auf das Singular-Handbuch [GPS] verwiesen):

- Polynomarithmetik (Addition und Multiplikation beliebiger Polynome, Division durch Monome), und zwar für Polynomringe über $\mathbb{Q}$, $\mathbb{F}_p$ mit $p \leq 32003$ Primzahl und über primitiven Erweiterungen (algebraisch oder transzendent) dieser Körper.

  `+, -, *, /`

- Substitution einer Ringvariablen $f(x,y) \mapsto f(x,g(x,y))$:

  `subst`

- Abbildungen zwischen Polynomringen über dem gleichen Körper:

  `map, imap, fetch`

- Abbildungen zwischen verschiedenen Koeffizientenkörpern $\mathbb{Q} \leftrightarrow \mathbb{F}_{32003}$, die natürliche Inklusionsabbildung $k(a) \hookrightarrow k(b)$, wobei $k(a) \subset k(b)$, $k = \mathbb{Q}$ oder $k = \mathbb{F}_p$ und $k(a)[x,y] \leftrightarrow k[x,y,z]$ k-linear, a algebraisch über k:

  `map`

- Größter gemeinsamer Teiler von ganzen Zahlen und bivariaten Polynomen über $\mathbb{Q}$, $\mathbb{F}_p$ und $\mathbb{F}_p(a)$, a algebraisch:

  `gcd`

- Erweiterter ggT von ganzen Zahlen, d. h. Berechnung von α, β, γ : γ ist ein ggT von a und b, und $a\alpha + b\beta = \gamma$

  `extgcd`

- Faktorisierung univariater Polynome über $\mathbb{Q}$, $\mathbb{F}_p$ und algebraischen Erweiterungen[1]:

  `factorize`

- Berechnung der ersten Syzygie (nur für den anfänglichen Test auf Quadratfreiheit nötig) und der Standardbasis eines Ideals (nur für die Berechnung des primitiven Elements):

  `syz, std, fglm`

- Partielle Ableitung multivariater Polynome (nur für Quadratfreiheitstest):

  `diff`

- Berechnung von Informationen über ein (im allg. bivariates) Polynom: Grad, Leitterm und dessen Grad, Leitkoeffizient, Leitexponent, (gewichteter) Jet und die einzelnen Koeffizienten:

  `deg, lead, ord, leadcoeff, leadexp, jet, coeffs`

- Berechnung von Informationen über den Polynomring, in dem sich alles abspielt, wie z. B. `charstr, ordstr, parstr, varstr, char, nvars, var, par, maxideal`

- Matrizen, Vektoren, Ideale und Listen, sowie damit zusammenhängende einfache Funktionen (z. B. Anzahl von Zeilen und Spalten):

  `matrix, intvec, ideal, list; size, ncols, nrows`

- Zufallszahlen (nur für die Berechnung des primitiven Elements):

  `randomid` aus der Bibliothek `random.lib`

[1] nach aktuellem Stand (Singular 1.3) für algebraische Erweiterungen von $\mathbb{Q}$ noch nicht vorhanden

5 Wichtige Prozeduren

Mit seinen über dreitausend Programmzeilen wäre ein vollständiges Listing der erstellten Bibliotheken zu umfangreich. Stattdessen werden wir uns auf die wichtigsten Prozeduren oder Ausschnitte daraus beschränken. Wir folgen dabei der natürlichen Reihenfolge, in der die Prozeduren gebraucht werden: Zuerst Tests, ob das Polynom quadratfrei ist, dann Prozeduren, die die Hamburger-Noether-Entwicklung berechnen und Hilfsprozeduren, die dazugehören, schließlich Prozeduren, die die fertig berechneten HNEs weiterverarbeiten. Die Prozeduren zur Berechnung eines primitiven Elements wurden in einer eigenen Bibliothek `primitiv.lib` zusammengefaßt, da sie vielleicht auch für andere Zwecke Verwendung finden können.

5.1 Test auf Quadratfreiheit

Zu Beginn beider HNE-Algorithmen (`HNE-irreduzibel` und `HNE-reduzibel`) müssen einige Tests durchgeführt werden, ob das übergebene Polynom zulässig ist. Die meisten sind kurz und deswegen in der Implementierung des Algorithmus selbst integriert (ist das Polynom bivariat? Ist es keine Einheit im Potenzreihenring?). Der Test, ob das Polynom eine quadratfreie Potenzreihe ist, ist dagegen etwas aufwendiger und in eine eigene Prozedur ausgelagert.

Zunächst sei noch einmal auf die Wichtigkeit dieses Tests verwiesen: Beide Algorithmen hängen entscheidend davon ab, daß ihre Eingabe quadratfrei ist, sonst terminieren sie nicht. Falls der Benutzer eine nicht quadratfreie Potenzreihe eingegeben hat, muß also entweder das Programm abgebrochen werden, oder die Eingabe muß durch einen garantiert quadratfreien Teiler ersetzt werden. Einen solchen Teiler liefert uns das Lemma 3.27: Sei $f \in k[x,y]$ oder $f \in k[\![x,y]\!]$, dann ist stets

$$l := \frac{f}{\mathrm{ggT}\,(f, \partial_x f, \partial_y f)}$$

ein quadratfreier Teiler von f, und außerdem gilt: Genau dann ist $l = f$, wenn f selbst schon quadratfrei ist.

Ist $f \in k[x,y]$, dann genügt es – wie schon im Korollar zu diesem Lemma gesehen – die Quadratfreiheit von f als Polynom sicherzustellen. Ist f quadratfrei in $k[x,y]$ und keine Einheit in $k[\![x,y]\!]$, dann ist $\operatorname{Spec} \overline{k}[\![x,y]\!]/\langle f\rangle$ eine ebene algebroide Kurve.

Falls der größte gemeinsame Teiler für k verfügbar ist, macht die Prozedur `squarefree` nichts anderes als l zu berechnen. Es gibt nur noch ein Problem: Der ggT im

Nenner von l ist im allgemeinen kein Monom (siehe Voraussetzungen des Systems). Da die Division garantiert aufgeht, können wir uns mit der ersten Syzygie behelfen:

$$l \cdot \operatorname{ggT}(f, \partial_x f, \partial_y f) - 1 \cdot f = 0 \quad \Longrightarrow \quad l = \texttt{syz}\left(\operatorname{ggT}(f, \partial_x f, \partial_y f), f\right)_1$$

In folgenden Körpern wollen wir die HNE ebenfalls berechnen, aber der ggT ist nicht implementiert: $\mathbb{Q}(a)$ und $\mathbb{F}_{p^k}$, wobei in letzterem Fall der Körper nicht durch ein algebraisches Element, sondern durch die Anzahl der Elemente gegeben ist.

Hier kommt `squarefree` mit der ersatzweisen Benutzung des `syz`-Kommandos zum Ziel: Sei $f' \in \{\partial_x f, \partial_y f\}$.

$$0 = f \cdot f' - f' \cdot f = \operatorname{ggT}(f, f') \left(\frac{f}{\operatorname{ggT}(f, f')} \cdot f' + \frac{-f'}{\operatorname{ggT}(f, f')} \cdot f \right)$$

$(f/\operatorname{ggT}(f, f'), -f'/\operatorname{ggT}(f, f'))$ ist also eine Syzygie von (f', f), und noch mehr: Sie erzeugt sogar den ersten Syzygienmodul.

$$\operatorname{ggT}(f, \partial_x f, \partial_y f) \bigm| \operatorname{ggT}(f, f') \quad \Longrightarrow \quad \texttt{syz}(f', f)_1 = \frac{f}{\operatorname{ggT}(f, f')} \biggm| l$$

Auf diese Weise erhalten wir folglich wenigstens einen Teiler von l, d. h.
$\texttt{syz}(f', f)_1$ ist auf jeden Fall ein quadratfreier Teiler von f, und
$\texttt{syz}(f', f)_1 = f \quad \Longrightarrow \quad f$ ist quadratfrei.

Bei f' haben wir noch die Freiheit, ob wir die partielle Ableitung von f nach x oder y nehmen wollen. Falls eine von beiden Null ist, können wir das ausnutzen und die andere Ableitung verwenden, denn mit 1 als quadratfreiem Teiler von f läßt sich natürlich nichts anfangen. Wenn allerdings beide partiellen Ableitungen Null sind, haben wir Pech gehabt; für diesen Fall siehe den folgenden Abschnitt „Problemfälle“.

```
proc squarefree (poly f)
"USAGE:  squarefree(f);  f a bivariate polynomial
RETURN: a squarefree divisor of f
        Normally the return value is a greatest squarefree divisor, but there
        is an exception: factors with a p-th root, p the ring characteristic,
        are lost.
EXAMPLE: example squarefree; shows some examples.
"
{
 //----------------- Wechsel in geeigneten Ring & Variablendefinition ---------
  def altring = basering;
  int e;
  int gcd_ok=1;
  string mipl="0";
  if (size(parstr(altring))==1) { mipl=string(minpoly); }
 //---- test: char = (p^k,a) (-> gcd not implemented) or (p,a) (gcd works) ----
  if ((char(basering)!=0) and (charstr(basering)!=string(char(basering)))) {
    string tststr=charstr(basering);
    tststr=tststr[1..find(tststr,",")-1];           //-> "p^k" bzw. "p"
    gcd_ok=(tststr==string(char(basering)));
```

```
 }
 execute("ring rsqrf = ("+charstr(altring)+"),(x,y),dp;");
 if ((gcd_ok!=0) && (mipl!="0")) { execute "minpoly="+mipl+";"; }
 poly f=fetch(altring,f);
 poly dif,g,l;
 if ((char(basering)==0) and (charstr(basering)!=string(char(basering)))
     and (mipl!="0")) {
   gcd_ok=0;                   // since Singular 1.2 gcd no longer implemented
 }
 if (gcd_ok!=0) {
//--------------------- Berechne f/ggT(f,df/dx,df/dy) ------------------------
   dif=diff(f,x);
   if (dif==0) { g=f; }        // zur Beschleunigung
   else { g=gcd(f,dif); }
   if (g!=1) {                 // sonst schon sicher, dass f quadratfrei
    dif=diff(f,y);
    if (dif!=0) { g=gcd(g,dif); }
   }
   if (g!=1) {
    e=0;
    if (g==f) { l=1; }         // zur Beschleunigung
    else {
      module m=syz(ideal(g,f));
      if (deg(m[2,1])>0) {
        "!! The Singular command 'syz' has returned a wrong result !!";
        l=1;                   // Division f/g muss aufgehen
      }
      else { l=m[1,1]; }
    }
   }
   else { e=1; }
 }
 else {
//------------------- Berechne syz(f,df/dx) oder syz(f,df/dy) ----------------
//-- Achtung: Ist f reduzibel, koennen Faktoren mit Ableitung Null verloren --
//-- gehen! Ist aber nicht weiter schlimm, weil char (p^k,a) nur im irred.  --
//-- Fall vorkommen kann. Wenn f nicht g^p ist, wird auf jeden Fall         --
//------------------------ ein Faktor gefunden. ------------------------------
   dif=diff(f,x);
   if (dif == 0) {
    dif=diff(f,y);
    if (dif==0) { e=2; l=1; } // f is of power divisible by char of basefield
    else { l=syz(ideal(dif,f))[1,1];  // x^p+y^(p-1) abgedeckt
           if (subst(f,x,0)==0) { l=l*x; }
           if (deg(l)==deg(f))  { e=1;}
           else {e=0;}
    }
   }
   else { l=syz(ideal(dif,f))[1,1];
          if (subst(f,y,0)==0) { l=l*y; }
          if (deg(l)==deg(f))  { e=1;}
          else {e=0;}
   }
 }
//--------------- Wechsel in alten Ring und Rueckgabe des Ergebnisses --------
 setring altring;
 if (e==1) { return(f); }    // zur Beschleunigung
```

```
  else {
   poly l=fetch(rsqrf,l);
   return(l);
  }
}
example
{ "EXAMPLE:"; echo = 2;
 ring exring=3,(x,y),dp;
 squarefree(x3+y);
 squarefree(x3);
 squarefree(x2);
 squarefree(xy+x);
 squarefree((x+y)^3*(x-y)^2); // (x+y)^3 is lost
 squarefree((x+y)^4*(x-y)^2); // result is (x+y)*(x-y)
}
```

Problemfälle

In den Fällen, wo der ggT verwendet werden kann, liefert `squarefree` bereits ein zufriedenstellendes Ergebnis. Wie schon bemerkt, wird nämlich von dieser Prozedur f genau dann zurückgegeben, wenn f quadratfrei ist, d. h. alle korrekten Eingaben f werden auch korrekt weiterverarbeitet. Es kann also nur Probleme geben, wenn f nicht quadratfrei ist.

$$\begin{array}{rll}
\text{Sei} \quad & k[x,y] \ni f = \prod\limits_i f_i^{q_i}, & f_i \text{ irreduzibel, paarweise verschieden,} \quad q_i > 0 \\
& \Longrightarrow & \partial_x f = \sum\limits_i q_i(\partial_x f_i) f_i^{q_i-1} \cdot \prod\limits_{j\neq i} f_j^{q_j}\,, \quad \partial_y f \text{ analog} \\
& \Longrightarrow & f_i^{q_i-1} \mid \mathrm{ggT}(f, \partial_x f, \partial_y f) \quad \forall\, i \\
& & f_i^{q_i} \mid \partial_z f \iff f_i \mid q_i\, \partial_z f_i \iff q_i\, \partial_z f_i = 0, \quad z \in \{x,y\} \\
\operatorname{char} k = 0 & \Longrightarrow & (f_i^{q_i} \nmid \partial_x f \ \vee\ f_i^{q_i} \nmid \partial_y f) \quad \forall\, i \\
& \Longrightarrow & l = \prod\limits_i f_i \text{ : das ist genau das, was wir haben wollen.} \\
\operatorname{char} k = p > 0 & \Longrightarrow & (\partial_x f_i \neq 0 \ \vee\ \partial_y f_i \neq 0) \quad \forall\, i, \quad \text{falls } k \text{ vollkommen ist.}
\end{array}$$

Beweis: Sei $\partial_x f_{i_0} = 0 = \partial_y f_{i_0} \quad \Longrightarrow \quad f_{i_0} = \sum\limits_{i,j} a_{i,j}\, x^{pi} y^{pj}$

Sei $\varphi \in \mathrm{End}(k)$ der Frobeniusendomorphismus, dann folgt aus der Vollkommenheit von k

$$f_{i_0} = \left(\sum_{i,j} \varphi^{-1}(a_{i,j}) x^i y^j \right)^p \quad ⚡ \text{ zu } f_{i_0} \text{ irreduzibel} \quad \blacksquare$$

Daraus folgt: $f_i^{q_i} \mid \mathrm{ggT}(f, \partial_x f, \partial_y f) \iff p \mid q_i$, und daraus wiederum

$$l = \prod_{p \nmid q_i} f_i \tag{5.1}$$

Wir verlieren also genau die Faktoren von f, die in einer Potenz vorkommen, die von der Charakteristik geteilt wird. Das bedeutet, daß schlimmstenfalls ein oder mehrere Zweige der Kurve von `HNE-reduzibel` nicht mehr gefunden werden, wenn man f durch l ersetzt.

Ist man damit nicht zufrieden, hat die Bibliothek `hnoether.lib` noch eine weitere Prozedur anzubieten, nämlich `allsquarefree`. Diese faktorisiert allerdings ein bivariates Polynom, was den Rechenaufwand stark erhöhen kann. Daher beschränken wir uns, wenn der Benutzer es nicht ausdrücklich anders wünscht, auf die Verwendung von `squarefree`. Für den Algorithmus `HNE-irreduzibel` ist `allsquarefree` darüberhinaus überflüssig, weil hier nur ein irreduzibles Polynom überhaupt zu einem Ergebnis führt.

Sinnvollerweise verwenden wir die Ergebnisse von `squarefree` weiter, da diese Prozedur in jedem Fall aufgerufen wird (um festzustellen, ob die Eingabe quadratfrei war). Bekannt sind also f und $l \in k[x,y]$ mit der Eigenschaft

$$f = \prod_i f_i^{q_i}\,, \qquad l = \prod_{p \nmid q_i} f_i$$

Lemma 5.1 *Definiere $g_j, h_j \in k[x,y]$ durch*

$$h_0 := f, \quad g_0 := l, \quad h_{j+1} := \frac{h_j}{g_j}, \quad g_{j+1} := \operatorname{ggT}(g_j, h_{j+1}), \quad j \geq 0$$

Dann gilt: Es gibt ein $j \in \mathbb{N}$, so daß $g_j = 1$ ist (oder assoziiert zu 1, da der ggT nicht eindeutig ist). Für dieses j gilt

$$h_j = \prod_{p \mid q_i} f_i^{q_i} \tag{5.2}$$

Beweis: Mit Induktion zeigt man, daß für alle $j \geq 0$ gilt:

$$\exists\, r_i^{(j)} \geq 0 : \quad h_j = \prod_{p \mid q_i} f_i^{q_i} \prod_{p \nmid q_i} f_i^{r_i^{(j)}}\,, \quad g_j = \prod_{p \nmid q_i,\, r_i^{(j)} > 0} f_i$$

Per definitionem folgt für alle i mit $p \nmid q_i$ aus $r_i^{(j)} > 0$, daß $r_i^{(j+1)} = r_i^{(j)} - 1$ ist. Daher existiert ein j mit $g_j = 1$, und zwar trifft das für alle $j \geq \max\left\{ r_i^{(0)} \,\middle|\, p \nmid q_i \right\} = \max\left\{ q_i \,\middle|\, p \nmid q_i \right\}$ zu.

Falls $g_j = 1$ ist, folgt $r_i^{(j)} = 0$ für alle i mit $p \nmid q_i$ und daraus die Behauptung. ▧

Wenn man dieses h_j aus (5.2) faktorisiert, bekommt man genau die noch fehlenden Faktoren, die man mit l aus (5.1) zu multiplizieren hat. Im Normalfall wird `allsquarefree` immer noch viel schneller sein als eine Faktorisierung des ursprünglichen f.

```
proc allsquarefree (poly f, poly l)
"USAGE : allsquarefree(f,l);  poly f,l
        l is the squarefree divisor of f you get by l=squarefree(f);
RETURN: the squarefree divisor of f consisting of all irreducible factors of f
NOTE  : This proc uses factorize to get the missing factors of f not in l and
        therefore may be slow.
EXAMPLE: example allsquarefree;  shows an example
"
{
 //------------------------ Wechsel in geeigneten Ring ------------------------
 def altring = basering;
 string mipl="0";
 if (size(parstr(altring))==1) { mipl=string(minpoly); }
 if ((char(basering)!=0) and (charstr(basering)!=string(char(basering)))) {
   string tststr=charstr(basering);
   tststr=tststr[1..find(tststr,",")-1];           //-> "p^k" bzw. "p"
   if (tststr!=string(char(basering))) {
     " Sorry -- not implemented for this ring (gcd doesn't work)";
     return(l);
   }
 }
 execute("ring rsqrf = ("+charstr(altring)+"),(x,y),dp;");
 if (mipl!="0") { execute "minpoly="+mipl+";"; }
 poly f=fetch(altring,f);
 poly l=fetch(altring,l);
 //---------- eliminiere bereits mit squarefree gefundene Faktoren ------------
 poly g=l;
 while (deg(g)!=0) {
   f=syz(ideal(g,f))[1,1];                         // f=f/g;
   g=gcd(f,l);
 }                                                 // jetzt f=h^p
 //--------------- Berechne uebrige Faktoren mit factorize --------------------
 if (deg(f)>0) {
  g=1;
  ideal factf=factorize(f,1);
  for (int i=1; i<=size(factf); i++) { g=g*factf[i]; }
  poly testp=squarefree(g);
  if (deg(testp)<deg(g)) {
    "!! factorize has not worked correctly !!";
    if (testp==1) {" We cannot proceed ..."; g=1;}
    else {" But we could recover some factors..."; g=testp;}
  }
  l=l*g;
 }
 //--------------- Wechsel in alten Ring und Rueckgabe des Ergebnisses --------
 setring altring;
 l=fetch(rsqrf,l);
 return(l);
}
example
{ "EXAMPLE:"; echo = 2;
  ring exring=7,(x,y),dp;
  poly f=(x+y)^7*(x-y)^8;
  poly l=squarefree(f);
  l; // x-y found because 7 does not divide 8
  allsquarefree(f,l); // all factors (x+y)*(x-y) found
}
```

Über allen Körpern, wo der ggT verfügbar ist, können wir also die Eingabe ohne Schwierigkeiten quadratfrei machen; mit `allsquarefree` gelingt dies sogar verlustfrei.

Wenn `squarefree` sich mit Syzygien behelfen muß, sind wir nur in der Lage, $f/\mathrm{ggT}(f, f')$ zu berechnen. Je nachdem, welche partielle Ableitung wir für f' gewählt haben, verlieren wir zusätzlich alle irreduziblen Faktoren, deren Ableitung nach x oder y Null ist. ($f_i^{q_i} \mid \partial_z f \iff q_i \, \partial_z f_i = 0$, s. o.) Das hat den unangenehmen Effekt, daß die Implikation „f quadratfrei $\Rightarrow$ `squarefree` gibt f zurück" nicht mehr gilt.

Dieser Fall tritt aber nur bei zwei Sorten von Körpern auf: Für Körper $k = \mathbb{F}_{p^l}$ ist sowieso nur `HNE-irreduzibel` implementiert. Das bedeutet, daß uns ein einziger irreduzibler Faktor bereits genügt. Da nicht beide partielle Ableitungen eines irreduziblen Polynoms Null sein können, bekommen wir mindestens einen irreduziblen Faktor, es sei denn, beide Ableitungen von f sind Null. Dann ist f p-te Potenz eines anderen Polynoms und wir begnügen uns damit, dem Benutzer dies mitzuteilen, denn für diesen Körper steht auch der Befehl `factorize` nicht zur Verfügung. Es bleibt aber immer noch die Möglichkeit, $\mathbb{F}_p(a)$ statt $\mathbb{F}_{p^l}$ zu benutzen.

Der zweite Typ von Körper, der zur Zeit noch ohne Implementation des ggT ist, ist $\mathbb{Q}(a)$, $a \in \mathbb{C}$ algebraisch. Weil diese Körper die Charakteristik Null haben, sind die einzigen Faktoren, die wir verlieren können, solche, die nur von x oder nur von y abhängen. Das sind bis auf Einheiten im Potenzreihenring lediglich x^n oder y^n. Falls $n = 0$ ist, haben wir gar keinen Zweig verloren, sondern die Eingabe sogar vereinfacht, ansonsten wurde der verlorengegangene Faktor in `squarefree` durch den `subst`-Befehl gefunden (vgl. Listing, ab Kommentar „`Berechne syz(f,df/dx) oder syz(f,df/dy)`").

5.2 Berechnung der Hamburger-Noether-Entwicklung

5.2.1 Transformationen und Newtonpolygon

Zur Berechnung der HNE haben wir im Teil I einige Transformationen eingeführt; diese wurden als eigene Prozeduren realisiert.

$$T(f,M,N)(x,y) := \frac{f\left(y, xy^Q\right)}{y^{QN}}, \quad M = QN + R$$

In leichter Abwandlung:

```
proc T_Transform (poly f, int Q, int N)
"USAGE:   T_Transform(f,Q,N);  f poly, Q,N int
RETURN:  f(y,xy^Q)/y^NQ   if x,y are the ring variables
NOTE:    this is intended for irreducible power series f
EXAMPLE: example T_Transform;  shows an example
"
{
 map T = basering,var(2),var(1)*var(2)^Q;
 return(T(f)/var(2)^(N*Q));
}
```

$$T_1(f,\delta,Q)(x,y) := f\left(x, y + \delta x^Q\right)$$

```
proc T1_Transform (poly f, number d, int Q)
"USAGE:   T1_Transform(f,d,Q);  f poly, d number, Q int
RETURN:  f(x,y+d*x^Q)   if x,y are the ring variables
EXAMPLE: example T1_Transform;  shows an example
"
{
 map T1 = basering,var(1),var(2)+d*var(1)^Q;
 return(T1(f));
}
```

$$T_2(f,\delta',M',N') := T_1\left(\frac{T^*(f,M',N')}{x^s y^t}, \delta', Q\right)$$

Da δ' beim Aufruf von `T2_Transform` noch nicht bekannt ist, nimmt die Prozedur δ statt δ' als Eingabe und gibt zusätzlich zu T_2 auch δ' zurück. s und t müssen ebenfalls erst noch berechnet werden. Dazu dividieren wir $T^*(f,M',N')$ so lange durch x bzw. y, wie die Division aufgeht. Um diese Division zu beschleunigen, machen wir uns die Tatsache zunutze, daß T^* einen Endomorphismus $\mathcal{T}^*$ auf $\mathbb{Z}^2$ induziert mit der Eigenschaft: Jeder Eckpunkt des Newtonpolygons $N(T^*(f))$ ist Bild eines Eckpunktes von $N(f)$ unter $\mathcal{T}^*$ (siehe auch das Bild auf Seite 76). Es kommen also die gleichen s,t heraus, wenn man f durch ein Polynom ersetzt mit gleichem Newtonpolygon wie f – wir nennen ein solches Polynom ein Referenzpolynom für f. Ein solches Referenzpolynom wird von der Prozedur ebenfalls als Parameter erwartet und darf normalerweise viel weniger Terme enthalten als f.

Da im implementierten Algorithmus das Newtonpolygon von f bereits bekannt ist, wenn **T2_Transform** aufgerufen wird, läßt sich ein Referenzpolynom (sogar mit minimaler Zahl von Monomen) ohne zusätzlichen Aufwand berechnen (dazu kann die Prozedur **referencepoly** benutzt werden). Sollte das nicht gehen (Aufruf z. B. nicht durch den Algorithmus), ist trotzdem mindestens ein Referenzpolynom von f bekannt, nämlich f selbst (dann gibt es natürlich keinen Zeitgewinn).

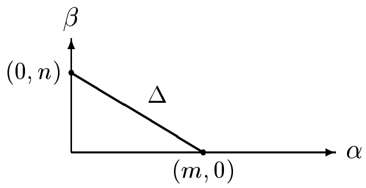
```
proc T2_Transform (poly f, number d, int M, int N, poly refpoly)
"USAGE:   T2_Transform(f,d,M,N,ref); f poly, d number; M,N int; ref poly
RETURN:  list: poly T2(f,d',M,N), number d' in \{ d, 1/d \}
ASSUME:  ref has the same Newton polygon as f (but can be simpler)
         for this you can e.g. use the proc `referencepoly' or simply f again
COMMENT: T2 is a composition of T_Transform and T1_Transform; the exact
         definition can be found in  Rybowicz: `Sur le calcul des places ...'
         or in  Lamm: `Hamburger-Noether-Entwicklung von Kurvensingularitaeten'
SEE ALSO: T_Transform, T1_Transform, referencepoly
EXAMPLE: example T2_Transform;  shows an example
"
{
 //---------------------- compute gcd and extgcd of N,M -----------------------
  int ggt=gcd(M,N);
  M=M/ggt; N=N/ggt;
  list ts=extgcd(M,N);
  int tau,sigma=ts[2],-ts[3];
  int s,t;
  poly xp=var(1);
  poly yp=var(2);
  poly hilf;
  if (sigma<0) { tau=-tau; sigma=-sigma;}
 // es gilt: 0<=tau<=N, 0<=sigma<=M, |N*sigma-M*tau| = 1 = ggT(M,N)
  if (N*sigma < M*tau) { d = 1/d; }
 //-------------------------- euklid. Algorithmus ----------------------------
  int R;
  int M1,N1=M,N;
  for ( R=M1%N1; R!=0; ) { M1=N1; N1=R; R=M1%N1;}
  int Q=M1 / N1;
  map T1 = basering,xp,yp+d*xp^Q;
  map Tstar=basering,xp^(N-Q*tau)*yp^tau,xp^(M-sigma*Q)*yp^sigma;
 //------------------- Durchfuehrung der Transformation T2 --------------------
  f=Tstar(f);
  refpoly=Tstar(refpoly);
 //--- dividiere f so lange durch x & y, wie die Div. aufgeht, benutze ein  ---
 //--- Referenzpolynom mit gleichem Newtonpolynom wie f zur Beschleunigung: ---
  for (hilf=refpoly/xp; hilf*xp==refpoly; hilf=refpoly/xp) {refpoly=hilf; s++;}
  for (hilf=refpoly/yp; hilf*yp==refpoly; hilf=refpoly/yp) {refpoly=hilf; t++;}
  f=f/(xp^s*yp^t);
  return(list(T1(f),d));
}
example
{ "EXAMPLE:"; echo = 2;
  ring exrg=0,(x,y),ds;
  export exrg;
  poly f=y2-2x2y+x6-x5y+x4y2;
  T2_Transform(f,1/2,4,1,f);
  T2_Transform(f,1/2,4,1,referencepoly(newtonpoly(f)));
  // if  size(referencepoly) << size(f)  the 2nd example would be faster
```

```
  referencepoly(newtonpoly(f));
  kill exrg;
}



proc referencepoly (list newton)
"USAGE:   referencepoly(newton);
         newton is list of intvec(x,y) which represents points in the Newton
         diagram (e.g. output of the proc newtonpoly)
RETURN:  a polynomial which has newton as Newton diagram
EXAMPLE: example referencepoly;  shows an example
"
{
 poly f;
 for (int i=1; i<=size(newton); i++) {
   f=f+var(1)^newton[i][1]*var(2)^newton[i][2];
 }
 return(f);
}
example
{ "EXAMPLE:"; echo = 2;
 ring exring=0,(x,y),ds;
 referencepoly(list(intvec(0,4),intvec(2,3),intvec(5,1),intvec(7,0)));
}
```

Das Newtonpolygon

Singular besitzt keine Funktion, um das Newtonpolygon zu berechnen, darum habe ich zwei Prozeduren geschrieben, die diese Aufgabe erledigen. Um die Rechenzeit zu verkürzen, sind sie auf bestimmte Polynome zugeschnitten:

1. Fall: f ist irreduzibel (oder wird als irreduzibel angenommen). Dann besteht $N(f)$ nach Satz 3.2 entweder aus einer Halbgeraden oder aus einer Seite. Es genügt also, die Schnittpunkte des Newtonpolygons mit den beiden Koordinatenachsen zu berechnen, was sich äußerst schnell durchführen läßt:




```
proc getnm (poly f)
"USAGE:   getnm(f); f bivariate polynomial
RETURN:  intvec(n,m) : (0,n) is the intersection point of the Newton
         polygon of f with the y-axis, n=-1 if it doesn't exist
         (m,0) is its intersection point with the x-axis,
         m=-1 if this point doesn't exist
ASSUME:  ring has ordering `ls' or `ds'
EXAMPLE: example getnm; shows an example
"
{
 // assume being called by develop ==> ring ordering is ls (ds would also work)
```

```
 return(ord(subst(f,var(1),0)),ord(subst(f,var(2),0)));
}
```

Da wir bei `HNE-irreduzibel` nicht wissen, ob f irreduzibel ist, gibt es eine Prozedur, die testet, ob das Newtonpolygon wirklich keine weiteren Eckpunkte besitzt:

```
proc testreducible (poly f, int n, int m)
"USAGE:   testreducible(f,n,m);  f poly, n,m int
RETURN:  1 if there are points in the Newton diagram below the line (0,n)-(m,0)
         0 else
EXAMPLE: example testreducible;  shows an example
"
{
 return(size(jet(f,m*n-1,intvec(n,m))) != 0)
}
```

2.Fall: f ist reduzibel. Nach der Generalvoraussetzung für den reduziblen Fall können wir annehmen, daß $N(f)$ Anfangs- und Endpunkt auf den Achsen hat.

Algorithmus `Newtonpolygon`

Eingabe: $f \in k[\![x,y]\!]$ mit $f(x,0) \neq 0 \neq f(0,y)$, $f(0,0) = 0$

Ausgabe: L Liste von Koordinaten der Eckpunkte von $N(f)$, aufsteigend nach der x-Koordinate sortiert (d. h. je zwei aufeinanderfolgende Punkte in L definieren eine Seite von $N(f)$)

Start

$A \leftarrow (0, \mathrm{ord}_y f(0,y))$, $\quad B \leftarrow (\mathrm{ord}_x f(x,0), 0)$, $\quad L \leftarrow A$
entferne alle Monome aus f, die in $D(f)$ auf oder oberhalb der Geraden $\overline{AB}$ liegen

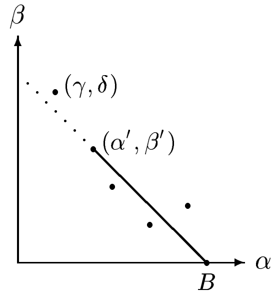
`solange` $f \neq 0$
  abbruch $\leftarrow$ Falsch
  `wiederhole`
    berechne α, β mit $x^\alpha y^\beta =$ Leitmonom von f bzgl. der revers lexikographischen Ordnung
    revlex: $1 > y > y^2 > \ldots > x > xy > \ldots$
    $C \leftarrow (\alpha, \beta)$
    `falls` es Punkte in $D(f)$ unterhalb der Geraden $\overline{AC}$ gibt
    `dann` entferne alle Monome $x^\gamma y^\delta$ aus f mit $\gamma \leq \alpha$
(a)   `sonst` *abbruch* $\leftarrow$ Wahr
  `bis` *abbruch*
(b) berechne α', β' mit $(\alpha', \beta') \in D(f) \cap \overline{AC}$ und β' minimal
  $A \leftarrow (\alpha', \beta')$, $\quad L \leftarrow L, A$
(c) entferne alle Monome aus f, die in $D(f)$ auf oder oberhalb der Geraden $\overline{AB}$ liegen

$L \leftarrow L, B$
`return` L

Ende

Verifikation: Der Algorithmus terminiert, denn mit Eintritt in die äußere Schleife besitzt f nur noch endlich viele Monome, und mit jedem Durchlauf (der äußeren Schleife, und bis auf den letzten Durchlauf auch der inneren) wird mindestens ein Monom von f entfernt. Zur Korrektheit des Ergebnisses:

- Jeder zurückgegebene Punkt ist Eckpunkt von $N(f)$: Für die außerhalb der Schleife berechneten Punkte A und B ist das offensichtlich, die anderen Punkte (α', β') gehören zur konvexen Hülle von $D(f)$ (wie alle Punkte von $\overline{AC} \cap D(f)$ wegen (a)). Weil β' minimal ist in (b), sind diese Punkte ebenfalls Eckpunkte.
- Jeder Eckpunkt von $N(f)$ wird zurückgegeben, denn es werden nur Punkte aus $D(f)$ entfernt, die keine Eckpunkte sind, oder die in L aufgenommen worden sind.
- Die Punkte in L sind wie gewünscht sortiert: Erster und letzter Punkt sind gerade die Schnittpunkte von $N(f)$ mit den Koordinatenachsen. Sei (α', β') ein in (b) berechneter Punkt. Angenommen es gebe ein Monom $x^\gamma y^\delta$ von f mit $\gamma \leq \alpha'$, welches nicht in (c) eliminiert wird, also unterhalb der Geraden durch (α', β') und B liegt. Dann befände sich (α', β') oberhalb der Strecke von (γ, δ) nach B, wäre also kein Eckpunkt von $N(f)$. Wir haben aber schon gezeigt, daß dieser Punkt ein Eckpunkt ist.

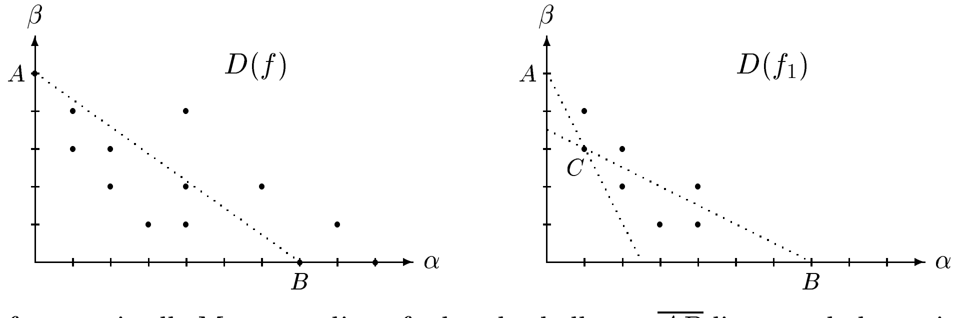
  

  Weil am Ende der Schleife folglich nur noch Monome $x^\gamma y^\delta$ von f übrig sind mit $\gamma > \alpha'$, gilt für die Koordinaten (γ', δ') des nächsten berechneten Eckpunktes: $\gamma' > \alpha'$. ▧

Beispiel: $f = \begin{cases} y^5 + 2xy^3 - xy^4 - 2x^2y^2 + x^2y^3 + x^3y + x^4y \\ -3x^4y^2 + 3x^4y^4 + x^6y^2 - x^7 + x^8y - x^9 \end{cases} \in \mathbb{Q}[[x, y]]$

Zur Vorbereitung berechnen wir

$$f(0, y) = y^5, \quad f(x, 0) = -x^7 - x^9 \quad \Longrightarrow \quad A = (0, 5), \quad B = (7, 0)$$



Entfernen wir alle Monome, die auf oder oberhalb von $\overline{AB}$ liegen, erhalten wir als Polynom zu Beginn der Schleife

$$f_1 = 2xy^3 - xy^4 - 2x^2y^2 + x^2y^3 + x^3y + x^4y - 3x^4y^2$$

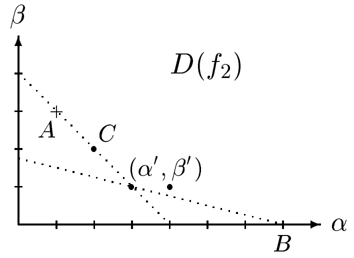
Das Leitmonom von f_1 bzgl. *revlex* ist xy^3, also ist $\quad C = (1,3)$

Es gibt keine Punkte unterhalb der Geraden $\overline{AC}$ (d. h. wir haben eine neue Seite von $N(f)$ gefunden mit Anfangspunkt A, der Endpunkt wird als nächstes bestimmt), also wird die innere Schleife nicht wiederholt.

$$D(f_1) \cap \overline{AC} = \{C\} \quad \Longrightarrow \quad (\alpha', \beta') = (1,3) =: A_{neu}, \quad L = [(0,5),(1,3)]$$

Nun entfernen wir alle Monome, die zu einem Punkt auf oder oberhalb $\overline{AB} = \overline{(1,3),(7,0)}$ in $D(f_1)$ gehören. Damit lautet das nächste zu betrachtende Polynom

$$f_2 = -2x^2y^2 + x^3y + x^4y$$



$$L(f_2) = x^2y^2 \quad \Longrightarrow \quad C = (2,2)$$

Wiederum gibt es keine Punkte unterhalb $\overline{AC}$.

$$D(f_2) \cap \overline{AC} = \{(2,2),(3,1)\}$$

$$\Longrightarrow \quad (\alpha', \beta') = (3,1) =: A_{neu}, \quad L = [(0,5),(1,3),(3,1)]$$

Alle Punkte in $D(f_2)$ liegen auf oder oberhalb von $\overline{AC} = \overline{(3,1),(7,0)}$, daher wäre das nächste zu betrachtende Polynom $\quad f_3 = 0, \quad$ d. h.

$$L = [(0,5),(1,3),(3,1),(7,0)]$$

ist die vollständige Liste der Eckpunkte von $N(f)$.

```
proc newtonpoly (poly f)
"USAGE:   newtonpoly(f);   f poly
RETURN:  list of intvec(x,y) of coordinates of the Newton polygon of f
ASSUME: For performance reasons, newtonpoly assumes that
        - the basering has ordering `ls'
        - f(x,0) != 0 != f(0,y), f(0,0) = 0
EXAMPLE: example newtonpoly;  shows an example
"
{
 intvec A=(0,ord(subst(f,var(1),0)));
 intvec B=(ord(subst(f,var(2),0)),0);
 intvec C,H; list L;
 int abbruch,i;
 poly hilf;

 L[1]=A;
 //-------- wirf alle Monome auf oder oberhalb der Geraden AB raus: -----------
 f=jet(f,A[2]*B[1]-1,intvec(A[2],B[1]));
 map xytausch=basering,var(2),var(1);
 for (i=2; f!=0; i++) {
   abbruch=0;
   while (abbruch==0) {
```

```
    C=leadexp(f);           // Ordnung ls ist wesentlich!
    if (jet(f,A[2]*C[1]-A[1]*C[2]-1,intvec(A[2]-C[2],C[1]-A[1]))==0)
      { abbruch=1; }        // keine Monome unterhalb der Geraden AC

// ----- alle Monome auf der Parallelen zur y-Achse durch C wegwerfen: -------
    else { f=jet(f,-C[1]-1,intvec(-1,0)); }
  }
//- finde alle Monome auf der Geraden durch A und C (unterhalb gibt's keine) -
  hilf=jet(f,A[2]*C[1]-A[1]*C[2],intvec(A[2]-C[2],C[1]-A[1]));

  H=leadexp(xytausch(hilf));
  A=H[2],H[1];
  L[i]=A;
//----------------- alle Monome auf oder unterhalb AB raus -------------------
  f=jet(f,A[2]*B[1]-1,intvec(A[2],B[1]-A[1]));
 }
 L[i]=B;
 return(L);
}
```

Wir kommen nun zu den eigentlichen Prozeduren, die Hamburger-Noether-Entwicklungen berechnen. Wenn

$$z_{j-1} = \sum_{i=1}^{h_j} a_{ji}\, z_j^i + z_j^{h_j} z_{j+1}\,, \quad j := 0, \ldots, r$$

eine HNE ist, sind wir an folgenden Daten interessiert:

(i) Die Hamburger-Noether-Matrix, bestehend aus den a_{ji}, wobei die Berechnung an irgendeiner Stelle in der letzten Zeile abgebrochen worden ist. Für $z_{-1} \neq 0$ soll mindestens ein $a_{ri} \neq 0$ bekannt sein, denn unter dieser Voraussetzung läßt sich die Multiplizitätensequenz der zugehörigen Kurve (vgl. Lemma 4.1) und damit jede Kurveninvariante berechnen.

(ii) Die Information, bei welchem Element a_{ri} die Berechnung abgebrochen wurde, oder daß die HN-Matrix vollständig berechnet ist (d. h. die HNE ist endlich).

(iii) Die h_j, $j := 0, \ldots, r-1$

(iv) Die Information, zu welchem Parameter z_0 und z_{-1} gehören; also falls $z_{-1}, z_0, \ldots, z_r$ eine HNE von $f \in k[[x,y]]$ ist, wollen wir wissen, ob dabei $z_0 = [x]_f$ oder $z_0 = [y]_f$ gilt.

(v) Schließlich ist es von Vorteil, falls die Berechnung der HNE abgebrochen wurde, die zuletzt transformierte Potenzreihe zu kennen. Dann hat man nämlich alle Informationen, um zur Ermittlung weiterer Terme in der letzten Zeile der HNE die Berechnung an der Stelle fortzusetzen, wo man aufgehört hat. So kann man z. B. nachträglich die Genauigkeit der Parametrisierung, die man aus der HNE erhält, beliebig erhöhen (vgl. Beispiel 2 zu `PrimParam-HNE`), ohne die HNE jedesmal von Anfang an neu berechnen zu müssen.

Diese Daten werden bei der Implementation in Singular in einer Liste `L` abgelegt, welche folgendes Format hat:

- Das erste Element `L[1]` ist vom Typ **`matrix`** und enthält die Hamburger-Noether-Matrix mit einer Ergänzung: Da die a_{ji} mit $i > h_j$ eigentlich gar nicht existieren (`L[1]` hat aber natürlich in jeder Zeile gleich viele Spalten), ist a_{j,h_j+1} auf die erste Variable des aktuellen Ringes gesetzt (normalerweise x) – dieser Eintrag markiert für den Betrachter das Ende der laufenden Zeile. Falls auch die letzte Zeile der HNE endlich ist, findet sich diese Markierung auch hier. Falls dagegen in der letzten Zeile undefinierte (weil nicht mehr berechnete) Werte auftreten, sind diese durch die zweite Ringvariable kenntlich gemacht (gewöhnlich y).

  `L[1]` enthält also schon alle wichtigen Informationen (i)–(iii), aus der man die HNE direkt ablesen kann. Die Ringvariablen in dieser Matrix sind dabei nur für den Betrachter eingefügt; für die Prozeduren, die `L` weiterverarbeiten, sind die Informationen (ii) und (iii) zur besseren Verfügbarkeit noch einmal explizit in `L[2]` abgelegt.

- `L[2]` ist vom Typ **`intvec`** und besteht aus $r+1$ Komponenten: Die ersten r sind $h_0, h_1, \ldots, h_{r-1}$, bei der letzten wurde $h_r (= \infty)$ ersetzt durch eine Zahl $z \in \mathbb{Z}$ mit folgender Bedeutung:

$z > 0$	signalisiert, daß die HNE endlich ist: $a_{ri} = 0 \quad \forall\, i > z$
$z = -1$	bedeutet, daß alle Einträge in der letzten Zeile von `L[1]` berechnet wurden. D. h. ist `L[1]` eine $m \times n$-Matrix, dann sind `L[1][m,1]` $= a_{r1}, \ldots,$ `L[1][m,n]` $= a_{rn}$ bekannt und die a_{ri} mit $i > n$ unbekannt.
$z < -1$	heißt, daß `L[1]` undefinierte Einträge haben kann, und zwar ist `L[1][m,-z]` die erste undefinierte Komponente (falls $-z \le n$); $a_{r,1}, \ldots, a_{r,-z-1}$ sind dagegen bekannt.

- `L[3]` ist vom Typ **`int`**; `L[3]` $= 0$ falls z_0 die Restklasse der ersten Ringvariable ist, sonst `L[3]` $= 1$.

- `L[4]` schließlich ist das zuletzt transformierte Polynom, falls eine Fortsetzung der Berechnung möglich (und gewünscht) ist. Aus Platzgründen (dieses Polynom kann sehr viel Speicher belegen) ist sonst `L[4]` $= 0$.

Beispiel: Der implementierte Algorithmus soll für $f = y^2 + x^7y + x^8 + x^9 \in \mathbb{F}_2[x,y]$ (vgl. Beispiel 1, S. 60) eines der folgenden Ergebnisse liefern:

$$\left[\begin{pmatrix} 0 & 0 & 0 & 1 & x \\ 0 & 1 & y & y & y \end{pmatrix},\ (4,-3),\ 0,\ y + x^2 + y^3 + xy^3\right]$$

$$\text{oder} \quad \left[\begin{pmatrix} 0 & 0 & 0 & 1 & x & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix},\ (4,-1),\ 0,\ f_3\right]$$

je nachdem, wann die Berechnung abgebrochen wurde.

5.2.2 Berechnung der HNE einer irreduziblen Kurve

Die Prozedur `develop` ist im Kern die direkte Umsetzung des Algorithmus `HNE-irreduzibel`. Dem vorgeschaltet sind diverse Tests, ob die Eingabe zulässig war:

(i) Der zugrundeliegende Polynomring muß mindestens zwei Variablen besitzen. Bei mehr Variablen wird zwar eine Warnung ausgegeben, aber weitergerechnet. Alle Variablen im übergebenen Polynom außer den ersten beiden werden durch Null ersetzt.

Desweiteren sind Ringe mit folgenden Einträgen für den Koeffizientenkörper unzulässig:

- `real`: Abbildungen zwischen solchen Ringen sind in Singular nicht implementiert, außerdem steht „`0`“ als Alternative zur Verfügung.
- (p,`a`) und (p,`b[,c,...]`), wobei p Primzahl, `a` $\in \overline{\mathbb{F}_p}$ algebraisch und `b,c,...` transzendent sind: Wir wollen im Koeffizientenkörper die p-te Wurzel ziehen – die Prozedur `develop` verwendet dazu aber einen Erzeuger der Einheitengruppe (siehe S. 58), der sich in diesen Ringen nicht so leicht ermitteln läßt, bzw. der gar nicht existiert.

  Auch hier gibt es Alternativen: `reddevelop` verwendet Faktorisierung und kommt darum mit diesen Ringen zurecht; für `develop` kann man im ersten Fall den gleichen Ring auch mit „(p^k,`e`)“ deklarieren, wobei p^k die Anzahl der Elemente von $\mathbb{F}_p(a)$ ist – mit dieser Deklaration kann `develop` arbeiten, denn jetzt ist `e` ein Erzeuger von $\mathbb{F}_p(a)^*$.

(ii) Als nächstes wird das übergebene Polynom f auf Zulässigkeit getestet. Dies beinhaltet den Test, ob f Null oder eine Einheit im Potenzreihenring ist. Falls eine der Ringvariablen f teilt, wird sofort die HNE $z_{-1} = 0$ als Ergebnis zurückgegeben. Sonst wird mit der Prozedur `squarefree` die Quadratfreiheit getestet.

Falls der Koeffizientenkörper $\mathbb{Q}$ ist, wird dieser Test zuerst in Charakteristik 32003 durchgeführt, um Rechenzeit zu sparen. Denn ist $\varphi : \mathbb{Q}[x,y] \to \mathbb{F}_p[x,y]$ die kanonische Abbildung und $f = g^2h$ nicht quadratfrei, dann trifft dies erst recht auf $\varphi(f) = \varphi(g)^2\varphi(h)$ zu.

Ist $\varphi(f)$ nicht quadratfrei, gibt es für den Benutzer mehrere Wahlmöglichkeiten, wie weiter verfahren werden soll; insbesondere kann der Test für f selbst wiederholt werden.

Ist f nicht quadratfrei, wird ein Hinweis ausgegeben und mit einem quadratfreien Teiler weitergerechnet, falls einer gefunden werden konnte (vgl. Abschnitt „Problemfälle“ in 5.1: Test auf Quadratfreiheit).

(iii) Schließlich wird geprüft, ob die Länge des Newtonpolygons $N(f)$ größer oder gleich der Höhe ist; wenn nicht, werden die beiden Variablen von f vertauscht, und wir merken uns dies (die einzige Konsequenz ist, daß bei der Rückgabe des Ergebnisses die ganze Zahl `L[3]` in der Liste 1 statt 0 ist).

Durch diese Modifikationen ist dafür gesorgt, daß `develop` für jedes beliebige übergebene Polynom terminiert und entweder eine HNE oder eine Fehlermeldung zurückgibt.

Die eigentliche Berechnung der HNE findet in einem einheitlichen Ring statt, und zwar werden ggf. die Variablen in x und y umbenannt und als Ordnung *revlex* $\widehat{=}$ `ls` gewählt (letzteres damit `ord` die richtigen Werte liefert). Falls der bei Aufruf der Prozedur aktuelle Ring diese Bedingungen nicht erfüllt, wird ein Ringwechsel durchgeführt, der am Ende wieder rückgängig gemacht wird.

Die Koeffizienten a_{ji} der HN-Matrix werden vorübergehend zeilenweise in Variablen vom Typ `ideal` gespeichert, da die Anzahl der benötigten Zeilen und Spalten der Matrix erst am Ende bekannt ist. Standardmäßig werden beide minimal gewählt, d. h. in der letzten Zeile werden gerade so viele Koeffizienten berechnet, daß es keine unbestimmten Einträge in der Matrix gibt und sich die Zahl der benötigten Spalten nicht unnötig erhöht. Weiteres dazu in der Dokumentation von `develop` im Anhang. Weil `HNE-irreduzibel` eine endliche HNE auch dann finden kann, wenn die Kurve in Wirklichkeit reduzibel ist (aber nur, wenn keiner der anderen Zweige bis zum Abbruch der Berechnung in seiner HNE abweicht), wird von `develop` ein weiteres Element an die zurückgegebene Liste `L` (wie auf S. 157 beschrieben) angehängt:

`L[5]` vom Typ `int` ist 1 wenn die Kurve genau einen Zweig hat (dessen HNE sich in `L[1]`, `L[2]`, `L[3]` befindet), sonst 0.

5.2.3 Berechnung der HNEs einer reduziblen Kurve

Die Prozedur `reddevelop` ist im Prinzip eine Implementation von `HNE-reduzibel` in der erweiterten Version von Seite 97. Dieser Algorithmus wurde allerdings für Singular in mehrfacher Hinsicht modifiziert:

- Die Rekursionstiefe in Singular ist beschränkt. Aus diesem Grund wurde darauf verzichtet, nach jeder Transformation T_2 `HN` und `HNΔ` neu aufzurufen. Stattdessen gibt es eine einzige Prozedur `HN`, die von `reddevelop` verwendet wird und sich nur dort selbst aufruft, wo der Auflösungsbaum eine echte Verzweigung hat. Im schlimmsten Fall ist die Rekursionstiefe also gleich der Zahl der Zweige, im Normalfall wächst sie logarithmisch mit der Zweigzahl der Kurve. Die HNE einer irreduziblen Kurve berechnet `reddevelop` im wesentlichen genauso wie `develop`.
- In Singular ist die Verarbeitung von Bäumen umständlich. Weil letzten Endes sowieso nur die Hamburger-Noether-Entwicklungen der einzelnen Zweige interessieren, wird intern nicht der Auflösungsbaum, sondern eine Liste in folgendem Format berechnet:

Definition 5.1 Eine **Entwicklungsliste** eines Polynoms $f \in k[x,y]$ ist eine Liste von Listen `L`, bei denen das erste Element vom Typ `intvec`, das letzte vom Typ `poly` und alle anderen vom Typ `ideal` sind, und so daß gilt:

Für jeden Zweig $\mathcal{Z}$ von $k[[x,y]]/\langle f\rangle$ gibt es genau eine Liste `L` (diese habe m Elemente) mit der Eigenschaft:

Falls `L[1][`$m-2$`]` > 0 ist, dann ist `L[`m`]` $= 0$ und

$$\begin{aligned} z_{j-2} &= \sum_{i=1}^{\mathtt{L}[1][j]} \mathtt{L}[j+1][i] z_{j-1}^i + z_{j-1}^{\mathtt{L}[1][j]} z_j \,, \quad j := 1,\ldots,m-3 \\ z_{m-4} &= \sum_{i=1}^{\mathtt{L}[1][m-2]} \mathtt{L}[m-1][i] z_{m-3}^i \end{aligned}$$

ist die (exakte!) Hamburger-Noether-Entwicklung von $\mathcal{Z}$.

Sonst ist `L[1][`$m-2$`]` < -1 und

$$\begin{aligned} z_{j-2} &= \sum_{i=1}^{\mathtt{L}[1][j]} \mathtt{L}[j+1][i] z_{j-1}^i + z_{j-1}^{\mathtt{L}[1][j]} z_j \,, \quad j := 1,\ldots,m-3 \\ z_{m-4} &= \sum_{i=1}^{-\mathtt{L}[1][m-2]} \mathtt{L}[m-1][i] z_{m-3}^i + \ldots \end{aligned}$$

ist der Anfang der HNE von $\mathcal{Z}$, deren Vervollständigung man erhält, wenn man `L[`m`]` entwickelt (dabei kommen keine weiteren Zeilen mehr hinzu).

Die Anzahl der Listen, aus denen eine Entwicklungsliste von f besteht, sei ferner gleich der Anzahl der Zweige der durch f definierten Kurve.

Bemerkung: Die Information, welcher Parameter für z_0 gewählt wurde, fehlt mit Absicht, denn wie beim Auflösungsbaum werden in einer Entwicklungsliste nur HNEs mit gleichem transversalen Parameter kodiert werden: entweder $z_0 = [x]_{f_i}$ für alle Faktoren f_i oder immer $z_0 = [y]_{f_i}$.

In Singular implementiert wurde nun eine zweigeteilte Prozedur `HN`; der erste Teil ist iterativ und berechnet die HNE so wie der auf Seite 68 modifizierte Algorithmus `HNE-irreduzibel`. Dabei gibt es drei Fälle:

1. Der Algorithmus stoppt, weil die HNE abbricht (`"Ende"`) oder weil festgestellt wird, daß die letzte Zeile der HNE erreicht ist (`"Abbruch"`).

2.a Der Algorithmus stoppt, weil das Newtonpolygon aus mehr als nur einer Seite besteht (`"f reduzibel!"`).

2.b Der Algorithmus stoppt, weil die quasihomogene Leitform nicht das Aussehen $c\left(y^{N'} - \delta x^{M'}\right)^e$ hat ($\Longleftrightarrow$ das charakteristische Polynom hat verschiedene Nullstellen) (`"f reduzibel!"`).

Im Fall 1 sind wir fertig – die Prozedur `HN` wird verlassen. Im Fall 2 tritt sowohl bei a) als auch bei b) eine Verzweigung auf. Es gibt mehrere HNEs, deren bisher berechneter Anfang identisch ist. Diese HNEs werden im zweiten Teil der Prozedur errechnet, welcher für jede Seite des Newtonpolygons und für jede Nullstelle der zugehörigen quasihomogenen Leitform `HN` rekursiv aufruft.

Bei 2.a ist zu beachten, daß nur der relevante Teil des Newtonpolygons betrachtet werden darf; das sind die Seiten, deren Abstand zur Abszisse eine vorgegebene Größe nicht überschreitet (vgl. **`HNE-1-Zweig`**). Aus diesem Grund genügt uns die einfache Prozedur **`getnm`** nicht; vielmehr muß das Newtonpolygon auch schon im ersten Teil mit **`newtonpoly`** berechnet, der irrelevante Teil abgeschnitten und dann erst geprüft werden, ob mehr als eine Seite übrig bleibt. Dagegen können wir uns im ersten Teil nach wie vor eine Faktorisierung des charakteristischen Polynoms sparen (und so vorgehen wie im irreduziblen Fall auf Seite 67 beschrieben: δ nach Formel (3.13) oder (3.14) berechnen und testen, ob das die einzige Nullstelle ist. Dies erledigt im reduziblen Fall die Prozedur **`factorfirst`**[1]).

Es folgt ein Überblick über die umformulierte Prozedur `HN`:

Funktion `HN`(f, I)

Parameter: $f \in k[\![x,y]\!]$ quadratfrei, $x \nmid f$, $y \nmid f$; $I \in \mathbb{N}$ s.d. alle Seiten mit Abstand[2] $\leq I$ von der Abszisse eine Steigung von mindestens -1 haben.

Rückgabe: Eine Entwicklungsliste aller Faktoren von f, deren Seite in $N(f)$ höchstens den Abstand I von der Abszisse hat.

Start

Teil2 $\leftarrow$ WAHR, $L \leftarrow ()$ // leere Liste

——— Erster Teil ———

`solange` das Newtonpolygon $N(f)$ unterhalb von I nur eine Seite Δ besitzt und ϕ_Δ nur eine Nullstelle δ

  $f_{alt} \leftarrow f$, $f \leftarrow T_2(f, \delta, M', N')$

  berechne aus (δ, M', N') die Zeilen $\mathtt{a}_i$ der HNE (vom Typ `ideal`) und den Vektor der $\mathtt{h}_i$ (vom Typ `intvec`)

  `falls` $y \,|\, f$

    `falls` $I = 1$

      markiere die letzte Zeile $\mathtt{a}_r$ als endlich,

      setze $\mathtt{h}_r$ auf $Q = M/N$

      `return list(list(h,`$\mathtt{a}_0, \ldots, \mathtt{a}_r$`,poly(0)))` und `STOP`

    `sonst` $f \leftarrow f_{alt}$, verlasse die Schleife

  `sonst`

    `falls` Höhe$(\Delta) = 1$

      Teil2 $\leftarrow$ FALSCH, verlasse die Schleife

——— Zweiter Teil ———

`falls` *Teil2*

  $\mathtt{L}' \leftarrow \mathtt{list}(\mathtt{h}, \mathtt{a}_0, \ldots, \mathtt{a}_r)$

[1] Es gibt eine Ausnahme: Wenn die p^l-te Wurzel über einer Erweiterung von $\mathbb{F}_p$ berechnet werden muß, gibt `factorfirst` auf und veranlaßt ebenfalls einen Sprung in den zweiten Teil, weil im reduziblen Fall kein Erzeuger der Einheitengruppe gegeben ist. Im zweiten Teil wird/werden die Nullstelle(n) dann durch Faktorisieren gefunden.

[2] gemeint ist $\beta \in \mathbb{N} : (\alpha, \beta)$ ist der obere Eckpunkt der Seite

```
        für jede Seite Δ von N(f) mit Abstand ≤ I von der Abszisse:
            solange φ_Δ nicht vollständig zerfällt
                wechsle in eine Körpererweiterung
            für jede Nullstelle δ_i von φ_Δ
                e_i ← Vielfachheit der Nullstelle δ_i,   L'' ← ()
                f' ← T_2(f, δ'_i, M', N')
                berechne aus (δ'_i, M', N') die Zeilen a_r, ..., a_s der HNE
                und h_r, ..., h_{s-1}
                falls  y | f'
                    L'' ← list(list(intvec(h_r, ..., h_{s-1}, Q),
                                          a_r, ..., a_s, poly(0)))
                    markiere die letzte HN-Zeile a_s in L'' als endlich
                    f' ← f'/y,    e_i ← e_i − 1
                falls  e_i > 0
                    L'' ← L'', HN(f', e_i)
                setze aus den Listen L' und L'' eine Entwicklungsliste
                    H zusammen
                L ← L, H
        return L
  sonst return list(list(h, a_0, ..., a_r, f))
Ende
```

Bemerkungen: Weil die Prozedur (zumindest bei nicht endlichen HNEs) nur im unverzweigten Teil abbricht, ist gewährleistet, daß jede der berechneten Hamburger-Noether-Entwicklungen sich von den anderen an mindestens einer Stelle unterscheidet (das ist wichtig für die Berechnung der Schnittmultiplizität, vgl. Satz 4.13). Die endlichen HNEs spielen dabei eine Sonderrolle: Ihre Berechnung kann nicht über den letzten von Null verschiedenen Koeffizienten a_{ri} hinaus fortgesetzt werden, auch wenn wir gerade in Teil 2 sind. Ferner gibt es möglicherweise einen weiteren Zweig, dessen HNE sich von der endlichen nur darin unterscheidet, daß sie hinter dem Koeffizienten a_{ri} noch weitergeht. Dort tritt also eine Verzweigung auf. Sind wir gerade im ersten Teil (d. h. $y \mid f$, $I > 1$), dann wird aus diesem Grund die letzte Transformation rückgängig gemacht und das Polynom in Teil 2 weiter behandelt.

Die Reihenfolge bei der Berechnung der Zweige ist die gleiche wie im ursprünglichen Algorithmus `HNE-reduzibel`, d. h. die Reihenfolge, in der die Entwicklungslisten zurückgegeben werden, braucht für die Berechnung der Multiplizitätensequenz (vgl. Bemerkung auf S. 130) nicht mehr geändert zu werden.

Zur Transformation T_2: Wenn M, N Länge und Höhe der ausgewählten Seite Δ sind, ist M sehr häufig ein Vielfaches von N (nämlich immer dann, wenn sich der nächste zu berechnende Koeffizient in der gleichen Zeile der HNE befindet wie der vorhergehende). In diesem Fall vereinfacht sich die Transformation zu

$$T_2(f, \delta, M', N') = T_1(f, \delta, M/N) = f\left(x, y + \delta x^{M/N}\right)$$

(vgl. Bemerkung zur Definition von T_2 im irreduziblen bzw. reduziblen Fall). Da sich dadurch einiges an Rechnung erübrigt, wird hier statt `T2_Transform` direkt `T1_Transform` aufgerufen.

Der Wechsel in eine Körpererweiterung hat sich unter Singular als recht aufwendig herausgestellt, was daran liegt, daß fast alle Daten an den Polynomring gebunden sind, in dem sie berechnet wurden. Wir gehen wie folgt vor:

Jedesmal wenn das charakteristische Polynom faktorisiert wird, werden die gefundenen Faktoren auf Linearität untersucht. Falls ein nichtlinearer Faktor gefunden wird, wird dieser an `splitring` weitergegeben; diese Prozedur wechselt den Polynomring in einen Ring über einer Körpererweiterung, über der dieser Faktor zerfällt (wohlgemerkt: in mindestens zwei nichttriviale Faktoren, aber nicht unbedingt vollständig!) Bevor wir nun fortfahren können, müssen wir erst sämtliche ringgebundenen Variablen neu deklarieren und ihren Inhalt in diesen neuen Ring transferieren. Dazu benötigen wir den alten Ring, der die Daten enthält, und eine Abbildung vom alten in den neuen. Als Namen für die Ringe wurde `EXTHNEring`(i), $i := 1, 2, \ldots$ gewählt – auf diese Weise ist es ein leichtes, an alle Ringe in der richtigen Reihenfolge heranzukommen (wir brauchen nicht nur die letzten beiden, wie sich gleich zeigen wird) – und die Abbildung erhalten wir, indem wir als optionalen Parameter das (bisherige) primitive Element a an `splitring` übergeben. Als Ergebnis liefert `splitring` den Wert $p(a)$, wobei a jetzt das neue primitive Element und $p(a)$ das alte ist. Die gesuchte Abbildung $\psi : k(\alpha)[x,y] \to k(\beta)[x,y]$ ist gegeben durch $\psi(x) = x$, $\psi(y) = y$, $\psi(\alpha) = p(\beta)(= \alpha)$ und ψ k-linear. Falls der ursprüngliche Körper keine Erweiterung war (sondern $\mathbb{Q}$ oder $\mathbb{F}_p$), entfällt dieser Parameter von `splitring`.

Wie schon angedeutet kann es sein, daß das charakteristische Polynom nach dem ersten Ringwechsel immer noch nicht vollständig zerfällt (ein weiterer Grund: es kann mehrere nichtlineare irreduzible Faktoren geben). Um die Rechenzeit zu verkürzen, wird darum zuerst nur das Polynom f abgebildet und das charakteristische Polynom erneut faktorisiert. Die bereits vorhandene teilweise Faktorisierung wird dabei weiterverwendet. Erst wenn ϕ_Δ vollständig in Linearfaktoren zerfällt, werden auch die übrigen Daten aus dem Ring, in dem sie zuletzt bekannt waren, übertragen. Die erst später definierten Ringe werden bis auf den aktuellen wieder gelöscht.

Das Ganze verkompliziert sich weiter durch die Rekursivität von `HN`: Jedesmal wenn `HN` verlassen wird, muß die aufrufende Prozedur überprüfen, ob seit dem Aufruf ein Ringwechsel stattgefunden hat. Falls dies der Fall ist, müssen auch hier wieder die Variablen neu deklariert und ihr Inhalt aus demjenigen Ring transferiert werden, der vor dem Aufruf aktiv war.[3] Das für die Abbildung benötigte Polynom p (s. o.) wird von `HN` als zweiter Rückgabewert neben der Entwicklungsliste geliefert (wurde im Überblick über `HN` weggelassen).

Bei der Zusammensetzung der Listen am Ende der inneren Schleife im zweiten Teil ist zu beachten, daß `HN` im allgemeinen eine Entwicklungsliste L'' mehrerer Zweige zum Ergebnis hat. Um daraus den Teil H der Entwicklungsliste L von

[3] Hier liegt der Hauptgrund, warum die früheren Ringerweiterungen nicht gelöscht werden können und alle einen eigenen Namen brauchen.

f zu konstruieren, muß die Liste L' des gemeinsamen Anfangs aller Elemente von H entsprechend oft dupliziert werden; die Zeilen $\mathtt{a}_0, \dots, \mathtt{a}_{r-1}$ aus L' bleiben unverändert, $\mathtt{a}_r, \dots, \mathtt{a}_{s-1}$ wurden aus der Wahl von Δ und δ_i hergeleitet, $\mathtt{a}_s$ wird mit der ersten Zeile der HNE des i-ten Zweiges von f' (das ist $L''[i][2]$) verknüpft, und die übrigen Zeilen werden unverändert aus $L''[i]$ übernommen. Analog wird der Vektor der Exponenten h_q (in $H[i][1]$) zusammengesetzt aus $L'[1]$, $h_r, \dots, h_{s-1}$ und $L''[i][1]$.

Die Prozedur `reddevelop` dient als Schnittstelle zwischen Benutzer und `HN`. Wie bei `develop` werden zuerst Tests gemacht, ob die Eingabe zulässig war. Zulässige Ringe müssen mindestens zwei Variablen besitzen (alle außer den ersten beiden werden ignoriert) und dürfen weder `real` noch `(`p^k`,a)` als Koeffizientenkörper haben, weil `factorize` über diesen nicht implementiert ist. Als Ersatz sind „`0`“ bzw. `(`p`,a)` mit Angabe des Minimalpolynoms verfügbar.

Ferner wird `reddevelop` durch transzendente Erweiterungen von $\mathbb{F}_p$ oder $\mathbb{Q}$ zu einer Warnung veranlaßt, die aber nicht zum Abbruch führt. Für diese Körper gibt es in Singular keine algebraischen Erweiterungen, d. h. der Algorithmus kann ausgeführt werden, solange kein Wechsel in eine Körpererweiterung nötig ist. `HN` berechnet in diesem Fall diejenigen Hamburger-Noether-Entwicklungen, die bereits über dem gegebenen Körper existieren (wenn wir Glück haben also alle).

Weil die HNE im allgemeinen nicht über dem ursprünglich gegebenen Polynomring existiert, wird von `reddevelop` in jedem Fall ein neuer Ring `HNEring` erzeugt, der nach dem Verlassen der Prozedur aktiv bleibt. Er hat wie der temporäre Ring von `develop` die Variablen x, y und die Ordnung `ls`.

Als nächstes wird geprüft, ob das eingegebene Polynom f zulässig ist, d. h. keine Einheit in $k[[x, y]]$, nicht Null und quadratfrei über $k[x, y]$ (letzteres zuerst über $\mathbb{F}_{32003}[x, y]$, falls $k = \mathbb{Q}$ ist). Stellt sich aufgrund des Ergebnisses von `squarefree` heraus, daß f nicht quadratfrei ist, gibt es drei Fälle:

- Falls `squarefree` über einem Körper der Charakteristik 0 aufgerufen wurde, hat es den maximalen quadratfreien Teiler von f als Ergebnis. In diesem Fall wird nur eine Warnung ausgegeben und mit diesem Teiler weitergerechnet.
- Falls $\operatorname{char} k = p > 0$ ist, sind die Faktoren verlorengegangen, die in p-ter Potenz oder einem Vielfachen von p in f vorkommen (vgl. S. 146). Der Benutzer wird (wenn die globale Variable `printlevel` > 0 ist) interaktiv vor die Wahl gestellt, die HNE für diesen Teiler (empfohlen) oder für den maximalen quadratfreien Teiler von f berechnen zu lassen (dieser wird von `allsquarefree` berechnet), oder die Berechnung abzubrechen (letzteres ist auch die Reaktion für `printlevel` ≤ 0).
- Falls $\operatorname{char} k = 0$ ist, `squarefree` jedoch über $\mathbb{F}_{32003}[x, y]$ aufgerufen wurde, ist das zurückgegebene Polynom auf jeden Fall quadratfrei (auch über k). Aber möglicherweise ist es über $k[x, y]$ gar kein Teiler von f, und möglicherweise ist f über $k[x, y]$ doch quadratfrei. Man kann also den Test über dem ursprünglichen Polynomring wiederholen (für `printlevel` ≤ 0 geschieht das

automatisch). Falls alle Koeffizienten von f zwischen -16001 und $+16001$ liegen und kein Exponent der Monome durch 32003 teilbar ist, wird angeboten, mit dem gefundenen „Faktor" weiterzurechnen – ist dem nicht so, ist die Wahrscheinlichkeit im ersten Fall zu gering, daß der „Faktor" wirklich ein Teiler von f ist, denn ein Polynom mit Koeffizienten außerhalb dieses Bereiches wird von der Komposition der in Singular implementierten Abbildungen $\mathbb{Q} \to \mathbb{F}_{32003} \to \mathbb{Q}$ nicht mehr auf sich selbst abgebildet. Ein Monom mit durch 32003 teilbarem Exponenten geht bei der entsprechenden Ableitung (vgl. Lemma 3.27) über $\mathbb{F}_{32003}$ fälschlicherweise verloren, daher muß man im zweiten Fall damit rechnen, daß das Ergebnis über $\mathbb{F}_{32003}$ erheblich vom Ergebnis über k abweicht (insbesondere daß f doch quadratfrei ist). Eine Wiederholung des Tests erscheint hier sinnvoller, als mit dem ersten Ergebnis weiterzurechnen.

Wie in Charakteristik p hat man auch in diesem Fall die Möglichkeit, die Berechnung abzubrechen; außerdem kann der Benutzer die Wiederholung des Tests über k verhindern, indem er versichert, daß das eingegebene Polynom wirklich quadratfrei ist. Bei dieser Option ist Vorsicht geboten: Die Verantwortung dafür, daß der Algorithmus terminiert, geht dabei auf den Benutzer über, denn sollte f doch nicht quadratfrei sein, kann das nicht mehr garantiert werden. Eine entsprechende Warnung wird erzeugt, bevor nun die eigentliche HNE-Berechnung beginnt.

Wie in `HNE-reduzibel` werden zuerst die Sonderfälle $y \mid f$ und $x \mid f$ abgearbeitet. Danach enthält das Newtonpolygon keine Halbgeraden mehr (f quadratfrei!) und kann mit `squarefree` berechnet werden. Der Eckpunkt (α, β), der die Seiten von $N(f)$, die zu Faktoren mit transversalem Parameter $[x]$ gehören, von den übrigen Seiten trennt, wird bestimmt (vgl. Schritt 2 von `HNE-reduzibel`), und `HN`$\big(f(x,y),\beta\big)$ und `HN`$\big(f(y,x),\alpha\big)$ werden aufgerufen (vgl. Schritt 3).

`HN` liefert die HNEs in Form einer Entwicklungsliste. Diese wird durch die Prozedur `extractHNEs` in eine Liste von HNEs konvertiert, die das auf Seite 157 beschriebene Format haben. Die HNEs von `HN`$\big(f(x,y),\beta\big)$ bekommen als dritten Eintrag `L[3]` eine Null (für $z_0 = [x]$), die anderen eine Eins (für $z_0 = [y]$).

Auch bei dieser Prozedur muß berücksichtigt werden, daß nach dem Verlassen von `HN` ein anderer Ring aktiv sein kann als beim Aufruf. Ist das der Fall, müssen wiederum einige Variable neu deklariert und ihr Inhalt restauriert werden. Außerdem übernimmt `reddevelop` in diesem Fall die Aufgabe aufzuräumen, d. h. der aktuelle Ring wird wieder `HNEring` genannt und die Ringe `EXTHNEring`(i) werden gelöscht, weil nicht mehr benötigt.

5.2.4 Fortsetzung einer teilweise berechneten HNE

Falls beim Aufblasen keine Stelle auftritt, wo die Hamburger-Noether-Entwicklung endet, brechen beide eben vorgestellten Prozeduren die Berechnung in der letzten Zeile ab und geben außer der HN-Matrix auch das zuletzt transformierte

Polynom zurück. Mit dieser Information läßt sich die HNE nachträglich fortsetzen, ohne die bisherige Rechnung noch einmal machen zu müssen. Im Vergleich zu `HNE-irreduzibel` und erst recht zu `HNE-reduzibel` haben wir es jetzt in mehrfacher Hinsicht einfacher:

- Es gibt immer genau eine zu betrachtende Seite (nämlich die unterste), und diese hat die Höhe 1 (das war gerade das Abbruchkriterium für beide Algorithmen).
- Aus diesem Grund brauchen nur noch Transformationen vom Typ T_1 gemacht zu werden; es gibt keinen Wechsel mehr in eine neue Zeile der HNE.
- Die Fortsetzung der HNE existiert vollständig im aktuell gewählten Körper, d. h. es sind keine Wechsel in eine Körpererweiterung mehr nötig.
- Denn die quasihomogene Leitform des transformierten Polynoms lautet

  $$L_\Delta(f_i) = cx^{M_-}(y - \delta x^M) \in \widetilde{k}[x,y] \quad \Longrightarrow \quad c, \delta \in \widetilde{k}$$

  wobei M_- für alle weiteren Transformationen konstant bleibt, 1 die Höhe und M die Länge der untersten Seite Δ von $N(f_i)$ ist. Das charakteristische Polynom ist also immer *linear*; δ ist ganz einfach zu berechnen.

Der Kern der Prozedur `extdevelop`, die für die Fortsetzung der HNE zuständig ist, sieht dementsprechend einfach aus:

```
int Q,M;
........
list Newton=newtonpoly(f);
int M1=Newton[size(Newton)-1][1];     // konstant
number delta;
if (Newton[size(Newton)-1][2]!=1) {
   " *** The transformed polynomial was not valid!!";}
else {
//--------------------- Fortsetzung der HNE ----------------------------------
 while (Q<Exaktheit) {
   M=ord(subst(f,y,0));
   Q=M-M1;
//------ quasihomogene Leitform ist c*x^M1*y+d*x^(M1+Q) => delta=-d/c: -------
   delta=-koeff(f,M,0)/koeff(f,M1,1);
   a[Q]=delta;
   "a("+string(zeile-1)+","+string(Q)+") =",delta;
   if (Q<Exaktheit) {
    f=T1_Transform(f,delta,Q);
    if (subst(f,y,0)==0) {
      "The HNE is finite!";
      a[Q+1]=x; Exaktheit=Q;
      f=0;                         // Speicherersparnis: f nicht mehr gebraucht
    }
   }
 }
}
```

Die Variable `Exaktheit` ist ein Parameter von `extdevelop` und bestimmt das neue Abbruchkriterium: Die Prozedur stoppt, wenn die HNE abbricht, oder sobald der Koeffizient $a_{r,\texttt{Exaktheit}}$ aus der HN-Matrix berechnet ist. Um diesen Kern herum gibt es wieder Ringwechsel (falls nötig) in einen temporären Ring und zurück, vorher weitere die Gültigkeit der Parameter betreffende Tests, und nachher werden die neu berechneten Koeffizienten aus `a` und das transformierte `f` in die HNE eingetragen.

5.3 Weiterverarbeitung der Hamburger-Noether-Entwicklung

Die Routinen in diesem Abschnitt erhalten als Parameter eine oder mehrere Hamburger-Noether-Entwicklungen und berechnen daraus weitere Informationen.

Die Prozedur `extdevelop`, die eine HNE als Eingabe hat und wieder eine (erweiterte) HNE zurückgibt, wurde schon im letzten Teil behandelt. Ferner gibt es Prozeduren zur Darstellung einer HNE auf dem Bildschirm und zur Berechnung der Anfangsterme einer Parametrisierung. Den größten Teil machen Prozeduren aus, die Kurveninvarianten berechnen.

5.3.1 Anzeige der Hamburger-Noether-Entwicklung

Gegeben ist eine HNE in der Form einer Liste wie auf Seite 157, gesucht ist das dazu korrespondierende Schema

$$
\begin{array}{rcl}
z_{-1} & = & a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 & = & a_{1,2}\, z_1^2 + a_{1,3}\, z_1^3 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2 \\
z_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{i-1} & = & a_{i,2}\, z_i^2 + a_{i,3}\, z_i^3 + \cdots\cdots + a_{i,h_i}\, z_i^{h_i} + z_i^{h_i} z_{i+1} \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{r-2} & = & a_{r-1,2}\, z_{r-1}^2 + \cdots + a_{r-1,h_{r-1}}\, z_{r-1}^{h_{r-1}} + z_{r-1}^{h_{r-1}} z_r \\
z_{r-1} & = & a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots\cdots\cdots\cdots\cdots\cdots
\end{array}
\qquad \text{(HN)}
$$

(vgl. Definition 2.1). Hier bieten sich zwei Möglichkeiten an: Entweder gibt die Prozedur zur Anzeige genau dieses Schema aus – aber nur auf dem Bildschirm – oder wir speichern dieses Schema zur weiteren Verarbeitung in einer Variable vom Typ `ideal` – dann müssen wir uns auf die rechte Seite beschränken. Da die linke Seite ohnehin immer gleich ist, habe ich mich für letzteres entschieden.

Das macht allerdings einen Ringwechsel unumgänglich: Es werden neue Variable $z_0, z_1, \ldots, z_r$ benötigt. Da der Benutzer aber die HNE normalerweise nur betrachten und in demjenigen Ring bleiben will, in dem die oben genannte Liste definiert ist, wird beim Beenden von `displayHNE` im Normalfall wieder in den alten Ring zurückgewechselt, was impliziert, daß das Ergebnis nicht zurückgegeben werden kann (es wird aber auf dem Bildschirm ausgegeben). Wird `displayHNE` dagegen mit einem optionalen zweiten Parameter aufgerufen, wird der neue Ring `displayring` beibehalten und der Rückgabewert der Prozedur ist die rechte Seite des Schemas (HN).

Da die HNE-Liste auch die Information enthält, welcher Parameter der Kurve zu welcher Zeile der HNE gehört, verarbeiten wir diese Daten ebenfalls; es gilt z. B.

$$\begin{aligned}
[y] = z_{-1} &= a_{0,1}\, z_0 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
[x] = z_0 &= a_{1,2}\, z_1^2 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2
\end{aligned}$$

oder die Plätze von $[x]$ und $[y]$ sind gerade vertauscht. Es folgt

$$\begin{aligned}
0 = z_{-1} - [y] &= -[y] + a_{0,1}\, z_0 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
0 = z_0 - [x] &= -[x] + a_{1,2}\, z_1^2 + \cdots\cdots + a_{1,h_1}\, z_1^{h_1} + z_1^{h_1} z_2
\end{aligned}$$

Die ersten beiden Komponenten des Ergebnisses wurden nun so modifiziert, daß sie die neuen rechten Seiten wiedergeben. Der erste Eintrag, den `displayHNE` zurückgibt, lautet also

$$-y + a_{0,1}\, z_0 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1$$

falls $z_{-1} = [y]$ ist, und sonst

$$-x + a_{0,1}\, z_0 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1$$

5.3.2 Berechnung einer Parametrisierung

Dies geschieht durch Rückwärtseinsetzen wie in Lemma 2.1 beschrieben (Prozedur `param`). Ist die HNE exakt, erhalten wir daraus eine Parametrisierung $(x(t), y(t))$ der Kurve. Wurde die Berechnung der HNE dagegen abgebrochen, parametrisiert $(x(t), y(t))$ eine andere Kurve mit endlicher HNE, die genau die bekannten Terme umfaßt.

Die Konsequenz ist, daß nur die Anfangsterme von $(x(t), y(t))$ mit der gesuchten Parametrisierung übereinstimmen; was noch fehlt ist eine Abschätzung, bis zu welchem Grad die Monome von $x(t)$ und $y(t)$ die korrekten Koeffizienten haben.

Lemma 5.2 *Es sei der Anfang*

$$\begin{aligned}
z_{-1} &= a_{0,1}\, z_0 + a_{0,2}\, z_0^2 + \cdots\cdots + a_{0,h_0}\, z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 &= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots &= \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \qquad\qquad (\mathsf{HN}) \\
z_{r-2} &= a_{r-1,2}\, z_{r-1}^2 + \cdots + a_{r-1,h_{r-1}}\, z_{r-1}^{h_{r-1}} + z_{r-1}^{h_{r-1}} z_r \\
z_{r-1} &= a_{r,2}\, z_r^2 + a_{r,3}\, z_r^3 + \cdots + a_{r,u-1}\, z_r^{u-1} + O(z_r^u)
\end{aligned}$$

der Hamburger-Noether-Entwicklung einer Kurve R gegeben.

Beim Rückwärtseinsetzen in (HN) *mögen wir die folgenden Potenzreihen erhalten:*

$$z_i(t) = c_{n_i} t^{n_i} + c_{n_i+1} t^{n_i+1} + \cdots + c_{u_i-1} t^{u_i-1} + O(t^{u_i}),\ c_{n_i} \neq 0, \quad i = -1, 0, \ldots, r-1$$

Dann sind die n_i und u_i eindeutig bestimmt durch

$$n_r = 1, \quad u_{r-1} = u \qquad \textit{und}$$

$$n_{i-1} = \begin{cases} n_i \cdot l \\ \\ n_i \cdot h_i + n_{i+1} \end{cases} \qquad u_{i-1} = \begin{cases} u_i + n_i(l-1) & \textit{falls } \exists\, 1 \leq l \leq h_i \\ & \textit{minimal, s. d. } a_{il} \neq 0 \\ u_i + n_i(h_i - 1) + n_{i+1} & \textit{falls } a_{il} = 0 \quad \forall\, l \end{cases}$$

Beweis: Die Formel für die n_i ist bereits aus Lemma 4.1 bekannt, da aus

$\underline{\nu}(z_r) = 1 = \operatorname{ord} z_r(t)$ (Korollar 1 zu Lemma 2.1) $\underline{\nu}(z_i) = \operatorname{ord} z_i(t) = n_i \ \forall\, i$

folgt. Offensichtlich ist auch der Wert von u_{r-1} korrekt.

Sei $0 \leq i < r$. Falls $z_{i-1} = a_{i,l}\, z_i^l + \ldots + a_{i,h_i}\, z_i^{h_i} + z_i^{h_i} z_{i+1}$ ist, $a_{i,l} \neq 0$, folgt

$$\begin{aligned} z_{i-1}(t) &= a_{i,l} \left(c_{n_i}^l t^{n_i l} + \ldots + O(t^{u_i}) t^{n_i(l-1)} + \ldots \right) + \ldots \\ &= c'_{n_i l} t^{n_i l} + \ldots + O\left(t^{u_i + n_i(l-1)} \right) \end{aligned}$$

Sonst gilt $z_{i-1} = z_i^{h_i} z_{i+1}$, und es folgt

$$\begin{aligned} z_{i-1}(t) &= \left(c_{n_i}^{h_i} t^{n_i h_i} + \ldots + O(t^{u_i}) t^{n_i(h_i-1)} \right) t^{n_{i+1}} \\ &= c'_{n_i h_i + n_{i+1}} t^{n_i h_i + n_{i+1}} + \ldots + O\left(t^{u_i + n_i(h_i-1) + n_{i+1}} \right) \end{aligned}$$

falls $i = r-1$ ist, sonst

$$\begin{aligned} z_{i-1}(t) &= \left(c_{n_i}^{h_i} t^{n_i h_i} + \ldots + O(t^{u_i}) t^{n_i(h_i-1)} \right) \left(c_{n_{i+1}} t^{n_{i+1}} + \ldots + O\left(t^{u_{i+1}}\right) \right) \\ &= c_{n_i}^{h_i} c_{n_{i+1}} t^{n_i h_i + n_{i+1}} + \ldots + O(t^{u_i}) t^{n_i(h_i-1)+n_{i+1}} + \ldots + O(t^{u_{i+1}}) t^{n_i h_i} \end{aligned}$$

Nach Induktionsvoraussetzung ist

$$u_i - n_i = \left\{ \begin{array}{c} u_{i+1} + n_{i+1}(l-1) - n_{i+1} l \qquad \text{oder} \\ u_{i+1} + n_{i+1}(h_{i+1} - 1) + n_{i+2} - n_{i+1} h_{i+1} - n_{i+2} \end{array} \right\} = u_{i+1} - n_{i+1}$$

$$\begin{aligned} &\Longrightarrow \quad u_{i+1} + n_i h_i - (u_i + n_i(h_i - 1) + n_{i+1}) = u_{i+1} - n_{i+1} - (u_i - n_i) = 0 \\ &\Longrightarrow \quad z_{i-1}(t) = c'_{n_i h_i + n_{i+1}} t^{n_i h_i + n_{i+1}} + \ldots + O\left(t^{u_i + n_i(h_i-1) + n_{i+1}} \right) \end{aligned}$$

▧

Korollar *Falls mindestens ein von Null verschiedener Koeffizient in der letzten Zeile von* (HN) *bekannt ist, haben $x(t)$ und $y(t)$ (durch Rückwärtseinsetzen in die unvollständige HNE berechnet) die gleichen Anfangsterme wie die „wirkliche" Parametrisierung von R. Genauer:*

Seien die Koeffizienten $a_{r,1},\dots,a_{r,u-1}$ *der letzten Zeile von* (HN) *bekannt, sei* $1 \le n \le u-1$ *minimal mit* $a_{r,n} \neq 0$, *sei*

$$\left(x(t) = t^{\mathrm{ord}(x)} \sum_i a_i\, t^i,\ y(t) = t^{\mathrm{ord}(y)} \sum_i b_i\, t^i \right)$$

die durch Rückwärtseinsetzen gewonnene Parametrisierung, dann besitzt R *eine Parametrisierung*

$$\left(t^{\mathrm{ord}(x)} \left(\sum_{i=0}^{u-n-1} a_i\, t^i + O(t^{u-n}) \right),\ t^{\mathrm{ord}(y)} \left(\sum_{i=0}^{u-n-1} b_i\, t^i + O(t^{u-n}) \right) \right)$$

Beweis: Falls $r > 0$ ist, gilt mit den Bezeichnungen des Lemmas 5.2 – wie im Beweis gesehen – für alle i:

$$u_i - n_i = u_{i+1} - n_{i+1}$$

Falls $r = 0$ ist, ist eine der Potenzreihen $x(t)$ oder $y(t)$ sogar exakt (nämlich t) und die Behauptung gilt erst recht. ▧

5.3.3 Berechnung von Kurveninvarianten

Viele Invarianten lassen sich günstiger aus anderen Kurveninvarianten berechnen als direkt aus der Hamburger-Noether-Entwicklung (man denke z. B. an die Puiseux-Paare). Die Berechnung der Invarianten irreduzibler Kurven wurde daher in der Prozedur `invariants` zusammengefaßt – so können aus den einmal berechneten Invarianten direkt die weiteren abgeleitet werden. Der Mehraufwand dafür, alle Invarianten auf einmal zu berechnen statt nur den gewünschten, ist völlig vernachlässigbar[4].

Folgendermaßen funktioniert `invariants`: Gegeben ist eine HNE in Gestalt einer Liste, wie sie von `develop` oder `extdevelop` zurückgegeben wird, oder eine Komponente der Rückgabe von `reddevelop`.

- Zuerst werden die Werte $n_i = \underline{\nu}(z_i), \quad i = 0,\dots,r$ ermittelt gemäß Lemma 4.1. Gemeinsam mit den $h_i, \quad i = 0,\dots,r-1$ erhalten wir daraus bereits die Multiplizitätensequenz der Kurve (bzw. des Zweiges).
- Parallel zu den n_i ergeben sich $s_0,\dots,s_g$ gemäß ihrer Definition in Gleichung (4.2) und daraus das Geschlecht g.
- Die charakteristischen Exponenten lassen sich nun mit der Formel aus Satz 4.7 berechnen:

$$\beta_0 = n_0, \quad \beta_{\nu+1} = \sum_{j=0}^{s_\nu} h_j n_j + n_{s_\nu} + n_{s_\nu+1} - n_0, \quad 0 \le \nu \le g-1 \qquad (4.5)$$

  Ausgehend von $s_0,\dots,s_g$ und $\beta_0,\dots,\beta_g$ werden weitere Invarianten ausgerechnet:

[4] andererseits reduziert sich der Aufwand, wenn man an mehr als nur einer Invarianten interessiert ist

- Die Puiseux-Paare $(m_1, n_1), \ldots, (m_g, n_g)$ nach Lemma 4.4 (iii):

$$m_1 = \frac{\beta_1}{\mathrm{ggT}(\beta_0, \beta_1)}, \quad m_i = \frac{n_1 \cdots n_{i-1}\beta_i}{\mathrm{ggT}(\beta_0, n_1 \cdots n_{i-1}\beta_i)}, \quad 2 \le i \le g$$

$$n_1 = \frac{\beta_0}{\mathrm{ggT}(\beta_0, \beta_1)}, \quad n_i = \frac{\beta_0}{\mathrm{ggT}(\beta_0, n_1 \cdots n_{i-1}\beta_i)}, \quad 2 \le i \le g$$

- Der Grad c des Konduktors nach Lemma 4.16:

$$c = \beta_g \left(n_{s_{g-1}} - 1\right) + \beta_{g-1}\left(n_{s_{g-2}} - n_{s_{g-1}}\right) + \ldots + \beta_1 (n_0 - n_{s_1}) - (n_0 - 1) \tag{4.15}$$

- Die δ-Invariante: $\delta = 2c$
- Die Erzeuger der Wertehalbgruppe lassen sich nach Lemma 4.15 (ii) direkt aus den Puiseux-Paaren ableiten:

$$\begin{aligned} \overline{\beta}_0 &= n_1 n_2 \cdots n_g, \quad \overline{\beta}_1 = m_1 n_2 \cdots n_g, \\ \overline{\beta}_i &= n_{i-1}\overline{\beta}_{i-1} + (m_i - m_{i-1}n_i)n_{i+1} \cdots n_g \end{aligned} \tag{4.11}$$

Die Rückgabe des Ergebnisses geschieht in einer Liste `L` mit den Einträgen

- Charakteristische Exponenten: $\texttt{L}[1] = \texttt{intvec}(\beta_0, \ldots, \beta_g)$
- Erzeuger der Halbgruppe: $\texttt{L}[2] = \texttt{intvec}(\overline{\beta}_0, \ldots, \overline{\beta}_g)$
- Puiseux-Paare: $\texttt{L}[3] = \texttt{intvec}(m_1, \ldots, m_g)$, $\texttt{L}[4] = \texttt{intvec}(n_1, \ldots, n_g)$
- Geschlecht: Nicht nötig, es gilt z. B. $g = \texttt{size}(\texttt{L}[3])$
- Grad des Konduktors: $\texttt{L}[5] = \texttt{int}(c)$
- δ-Invariante: Nicht nötig, $\delta = 2 \cdot \texttt{L}[5]$
- Multiplizitätensequenz: $\texttt{L}[6] = \texttt{intvec}(n_0, \ldots, n_0, n_1, \ldots\ldots, n_{r-1}, 1)$

Zur Anzeige der Invarianten auf dem Bildschirm existiert auch `displayInvariants`, womit die Notwendigkeit entfällt, sich die Bedeutung der zurückgegebenen Werte zu merken. Diese Prozedur verarbeitet auch die Liste der HNEs einer reduziblen Kurve und zeigt die Invarianten aller Zweige sowie eine Tabelle der Schnittmultiplizitäten an.

Es bleiben noch die Invarianten reduzibler Kurven, die von `separateHNE`, `intersection` und `multsequence` berechnet werden:

`separateHNE` berechnet die Anzahl der benötigten quadratischen Transformationen, um zwei Zweige einer reduziblen Kurve voneinander zu trennen, und wird insbesondere von `multsequence` weiterverwendet. Wenn $O_1, \ldots, O_N$ die beiden Zweigen gemeinsamen unendlich benachbarten Punkte sind, liefert `separateHNE` also den Wert $N+1$. Somit erhält man mit dieser Prozedur auch die Anzahl der gemeinsamen Punkte O_j. Das Ergebnis $N+1$ wird nach der Formel aus Satz 4.13 berechnet (Definition von s, i siehe dort):

$$N+1=\sum_{j=0}^{s-1} h_j + i \qquad \text{oder} \qquad N+1=1$$

Die erste Formel findet genau dann Anwendung, wenn für beide Kurven der gleiche transversale Parameter gewählt wurde. Weil `develop` und `reddevelop` – wenn möglich – immer $[x]$ als transversalen Parameter nehmen, folgt aus der Nichterfüllung dieser Bedingung automatisch die Gültigkeit der zweiten Formel (beide Zweige schneiden sich transversal).

Der gleiche Satz liefert auch die vier Formeln für die Schnittmultiplizität zweier Zweige, die `intersection` benutzt. Aus dem gleichen Grund wie oben tritt der Fall, daß keine von ihnen gültig ist, nie ein.

`multsequence` schließlich berechnet die Multiplizitätensequenz sowohl von irreduziblen als auch von reduziblen Kurven. Wird als Parameter nur eine HNE übergeben, erfolgt die Berechnung genauso wie bei `invariants` beschrieben (Lemma 4.1). Wird die Prozedur mit einer Liste von HNEs aufgerufen, erhalten wir das gewünschte Ergebnis durch Umsetzung des auf Seite 130 beschriebenen Verfahrens (Bemerkung zu Satz 4.13). Wichtig ist, daß die HNEs in dieser Liste eine geeignete Reihenfolge haben – wurde die Liste von `reddevelop` erzeugt, ist das gewährleistet.

Das Ergebnis von `multsequence` ist im reduziblen Fall eine Liste von zwei Werten vom Typ `intmat`: Die Spalten der ersten Matrix enthalten die Multiplizitätensequenzen der einzelnen Zweige; die i-te Zeile der zweiten Matrix sagt aus, welche Werte in der i-ten Zeile der ersten Matrix zu Zweigen mit gemeinsamem unendlich benachbarten Punkt in einer Umgebung $(i-1)$-ter Ordnung gehören.

Beispiel: Im Anschluß an die Bemerkung auf Seite 130 haben wir die Multiplizitätensequenz der durch $f := x(y^2+x^3y+x^3)(y^2+x^5y+x^3)(y^5+x^2) \in \overline{\mathbb{F}_2}[\![x,y]\!]$ gegebenen Kurve berechnet. `multsequence` gibt für diese Kurve folgende zwei Matrizen aus:

$$\begin{pmatrix} 2&2&1&2\\ 1&1&1&2\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1\\ 1&1&1&1 \end{pmatrix} \quad \text{und} \quad \begin{pmatrix} 4&0&0&0\\ 2&2&0&0\\ 2&2&0&0\\ 2&1&1&0\\ 2&1&1&0\\ 2&1&1&0\\ 1&1&1&1 \end{pmatrix} \quad \widehat{=} \quad \begin{array}{l} [(2,2,1,2)]\\ [(1,1),(1,2)]\\ [(1,1),(1,1)]\\ [(1,1),(1),(1)]\\ [(1,1),(1),(1)]\\ [(1,1),(1),(1)]\\ [(1),(1),(1),(1)]\\ \quad\cdots\cdots\cdots\cdots \end{array}$$

Die entsprechende Sequenz auf der rechten Seite kann mit der Prozedur `display-Multsequence` auf dem Bildschirm angezeigt werden.

5.4 Implementation der Körpererweiterung

Hier haben wir uns vor allem um die Berechnung des primitiven Elements zu kümmern. Ist das dazugehörige Minimalpolynom gefunden, kann dieses dazu benutzt werden, eine Ringerweiterung in Singular zu definieren. Die entsprechenden Routinen finden sich in einer eigenen Bibliothek `primitiv.lib`.

Es gibt zwei Prozeduren zur Berechnung des primitiven Elements: `primitive` ist eine Implementation von `Primitives Element` (vgl. S. 104) und `primitive_extra` eine von `Systematische Suche`.

Folgende Ergänzungen gibt es in `primitive` gegenüber dem auf Seite 104 dargelegten Algorithmus:

- Es wird gewährleistet, daß der Algorithmus in jedem Fall terminiert, indem nur endlich viele Elemente als primitives Element getestet werden (abhängig von der Mächtigkeit des Körpers).
- In Schritt 1 wird als erstes versucht, alle Koeffizienten $a_i = 0$ zu wählen – das entspricht einem Test, ob das letzte gegebene Element α_n der Körpererweiterung $k(\alpha_1, \ldots, \alpha_n)$ selbst schon primitiv ist.[5]
- Falls der Grundkörper k weniger als 60 Elemente enthält und $n = 2$ ist (d. h. es sind nur zwei algebraische Elemente α_1, α_2 gegeben – der wohl häufigste Fall, insbesondere für die Berechnung der HNE immer), wird a_1 nicht per Zufallsgenerator bestimmt, sondern geht systematisch die Elemente von k durch (steigende Wahrscheinlichkeit, daß das gleiche Element sonst mehrmals getestet wird).

  Andernfalls werden die a_i erst zufällig aus $\{-10, -9, \ldots, 9, 10\}$ gewählt, später $-100 \leq a_i \leq 100$, erst nach 20 Fehlversuchen $-10^8 \leq a_i \leq 10^8$ (für noch größere Zahlen liefert `random` keine befriedigende Streuung mehr). Dahinter steht das Bestreben, die Koeffizienten des resultierenden Minimalpolynoms klein zu halten (die v.a. über $\mathbb{Q}$ die Rechnung in der späteren Körpererweiterung sonst unnötig verlangsamen könnten).
- Schritt 2 erfordert die Berechnung einer reduzierten Standardbasis bzgl. der lexikographischen Ordnung. Dafür bietet sich der sogenannte FGLM-Algorithmus an, d. h. wir berechnen zuerst eine Standardbasis bezüglich *degrevlex*, der grad-revers-lexikographischen Ordnung (in Singular: `dp`), was deutlich schneller geht. Diese wird anschließend mit dem Befehl `fglm` in die eigentlich gesuchte Standardbasis umgewandelt.
- Bedingt durch die begrenzte Anzahl der Tests, aber wahrscheinlicher darum, weil keine der möglichen Linearkombinationen der α_i primitiv ist, kann es sein, daß `primitive` kein primitives Element findet. In diesem Fall wird als Ergebnis Null zurückgegeben.

[5] Auch in diesem Fall haben wir noch etwas zu berechnen: Wir kennen erst das Minimalpolynom von α_n über $k(\alpha_1, \ldots, \alpha_{n-1})$, wollen aber das Minimalpolynom über k

Bei `primitive_extra` müssen wir nur noch auf die Reihenfolge eingehen, in der die Koeffizienten a_{ij} von g gewählt werden, falls der Körper endlich ist.

Zunächst sei bemerkt, daß der Grad des berechneten Minimalpolynoms feststeht (unabhängig von der Wahl der a_{ij}; vgl. Satz 3.32 (ii)). Da die Koeffizienten nicht größer als die Charakteristik werden können und in der Praxis vorwiegend Körpererweiterungen niedriger Grade auftreten, spielt die Reihenfolge bei der Wahl keine große Rolle. Um auch die Standardbasenberechnung schnell zu machen (wieder mit `fglm`), werden zuerst die Polynome $g = z + ax, \quad a = 0, 1, \ldots, p-1$ getestet, dann $g = z + ax + bx^2$ u.s.w., d. h. Polynome niedrigen Grades werden bevorzugt, und unter diesen wiederum solche mit kleinen Koeffizienten. Erst wenn alle Versuche mit Polynomen der Form $z + f(x)$ erfolglos waren, werden auch Monome $x^i z^j, \quad i > 0$ oder $j > 1$ hinzugenommen (offensichtlich können konstante Terme entfallen, weil $g(\alpha, \beta)$ primitiv $\iff g(\alpha, \beta) + a$ primitiv $\forall\, a \in k$).

Nun kommen wir zum Höhepunkt von `primitiv.lib`, der Prozedur `splitring` zum komfortablen Wechsel in eine Körpererweiterung. Nur sie wird von `hnoether.lib` direkt benutzt.

Ausgangssituation: Es ist ein Polynomring r über K definiert, $K = k$ oder $K = k(\alpha)$ algebraische Erweiterung von $k \in \{\mathbb{Q}, \mathbb{F}_p\}$, und es ist ein beliebiges irreduzibles univariates Polynom $f \in r$ gegeben. Dann erzeugt `splitring` einen neuen Polynomring R über einer Körpererweiterung $\widetilde{K}$ von K, die mindestens eine Nullstelle des Polynoms enthält (falls es nicht bereits linear war, ist es über R also reduzibel). Nach Verlassen der Routine ist R der aktive Ring. Variablen und Monomordnung von r werden übernommen, der Name von R wird zusammen mit dem Polynom f an die Prozedur übergeben. Optional kann auch auf die Angabe eines neuen Namens verzichtet werden, was zur Folge hat, daß `splitring` den Ring r löscht und durch R (mit gleichem Namen) ersetzt. Schließlich ist es noch möglich, eine Liste L mit Elementen aus r als dritten Parameter zu übergeben; das Ergebnis der Prozedur ist dann die durch die Inklusion nach R abgebildete Liste.

Zur Implementation: Falls $K = k$ ist, ist die Aufgabe schnell erledigt: Wir suchen uns einen freien Namen `a` für das primitive Element und definieren

```
ring R = (p,a),...;    minpoly = f(a);
```

L kann einfach durch die identische Abbildung nach R transferiert werden.

Sonst werden f und das Minimalpolynom von r aufbereitet, um mit ihnen `primitive` aufzurufen und so das Minimalpolynom eines primitiven Elements von $\widetilde{K}$ zu erhalten. Sollte `primitive` kein solches finden, behebt `splitring` das Problem mit dem Aufruf von `primitive_extra`[6]. Der neu zu definierende Ring R unterscheidet sich von r lediglich durch das neue Minimalpolynom (sinnvollerweise übernehmen wir den Namen `a` des Parameters).

[6] Diese Prozedur ist langsamer, daher wird auch bei endlichen Körpern immer zuerst `primitive` versucht. Das schlägt meiner Erfahrung nach selbst in Charakteristik 2, 3 extrem selten fehl – ich habe noch kein schlechtes Beispiel gefunden.

Die Liste L kann jetzt nicht mehr mit der Identität abgebildet werden, weil der Parameter `a` in r und R verschiedene Elemente bezeichnet. Aber sowohl `primitiv` als auch `primitive_extra` berechnen ein Polynom `transform`, das

$$\texttt{transform}(\texttt{a}_{\in R}) = \texttt{a}_{\in r}$$

erfüllt. Das korrekte Aussehen von L in R erhalten wir folglich, wenn wir überall `a` durch `transform(a)` ersetzen, während alle Koefizienten aus k und Variablen gleich bleiben.

Dazu wechseln wir in einen Polynomring über k, der gegenüber r und R `a` als zusätzliche *Variable* hat (Abbildungen in Polynomringen sind in Singular zwangsläufig auf dem Koeffizientenkörper die Identität, daher muß `a` seine Rolle ändern)

$$\texttt{ring Z = } p\texttt{,(a,}\ldots\texttt{),}\ldots\texttt{;}$$

bilden L mittels `imap` in diesen Ring ab und wandeln es um mit der Abbildung, die definiert ist durch

```
ideal convid = maxideal(1);
convid[1] = transform;
map convert = basering,convid;
```

Das transformierte L wird schließlich mit `imap` nach R kopiert.

6 Analyse der Performance

Nachdem die Frage geklärt ist, *wie* die Berechnung der Hamburger-Noether-Entwicklung in Singular implementiert wurde, untersuchen wir nun, wie gut diese Implementation verschiedene Kurven entwickelt; das bedeutet in erster Linie *wie schnell*, als weiteres Kriterium dient auch der Speicherbedarf. Zuerst stellen wir einige Beispiele vor, an denen die Bibliothek `hnoether.lib` getestet wurde und betrachten anhand der größeren von ihnen, welche Teilschritte (prozentual) die aufwendigsten sind. Danach gehen wir auf einige der Schritte näher ein und sehen, welche Verbesserungen erzielt werden konnten, um zu dem im ersten Teil dargestellten Ergebnis zu kommen.

6.1 Rechenzeiten für verschiedene Beispiele

Dieser Abschnitt spiegelt die aktuelle Situation wider, d. h. die folgenden Tabellen enthalten Testergebnisse, die sich auf die Endversionen der Bibliotheken `hnoether.lib` und `primitiv.lib` und auf Singular in der Version 1.3.0 beziehen. Alle Rechenzeiten wurden auf einer HP-UX 9000/735/125 MHz Workstation ermittelt.

6.1.1 Irreduzible Beispiele

Hier geht es um die Performance der Prozedur `develop`. Natürlich kann man die gleichen Beispiele auch mit `reddevelop` behandeln – dies tun wir aber erst im nächsten Abschnitt.

Folgende Polynome (von Michel Merle) definieren (zumindest in Charakteristik 0, 2, 101 und 32003) irreduzible Kurvensingularitäten mit einer wachsenden Anzahl von Puiseux-Paaren:

	Puiseux-Paare
$p_1 := y^2 + x^3$	$(3,2)$
$p_2 := p_1^2 + x^5 y$	$(3,2),(7,2)$
$p_3 := p_2^2 + x^{10} p_1$	$(3,2),(7,2),(15,2)$
$p_4 := p_3^2 + x^{20} p_2$	$(3,2),(7,2),(15,2),(31,2)$
$p_5 := p_4^2 + x^{40} p_3$	$(3,2),(7,2),(15,2),(31,2),(63,2)$
$p_6 := p_5^2 + x^{80} p_4$	$(3,2),(7,2),(15,2),(31,2),(63,2),(127,2)$
$p_7 := p_6^2 + x^{160} p_5$	$(3,2),(7,2),(15,2),(31,2),(63,2),(127,2),(255,2)$

Tabelle 6.1: HNE von $p_1, \dots, p_6$ in Charakteristik 0, 2, 101 und 32003

$p_1: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 \end{aligned}$	$p_2: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 + z_1^2 z_2 \\ z_1 &= -z_2^2 - 2z_2^3 - 3z_2^4 + O(z_2^5) \end{aligned}$
$p_3: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 + z_1^2 z_2 \\ z_1 &= -z_2^2 + z_2^2 z_3 \\ z_2 &= -z_3^2 + 2z_3^3 - 7z_3^4 + O(z_3^5) \end{aligned}$	$p_4: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 + z_1^2 z_2 \\ z_1 &= -z_2^2 + z_2^2 z_3 \\ z_2 &= -z_3^2 + z_3^2 z_4 \\ z_3 &= -z_4^2 - 6z_4^3 - 29z_4^4 + O(z_4^5) \end{aligned}$
$p_5: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 + z_1^2 z_2 \\ z_1 &= -z_2^2 + z_2^2 z_3 \\ z_2 &= -z_3^2 + z_3^2 z_4 \\ z_3 &= -z_4^2 + z_4^2 z_5 \\ z_4 &= -z_5^2 + 10z_5^3 - 145z_5^4 + O(z_5^5) \end{aligned}$	$p_6: \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= -z_1^2 + z_1^2 z_2 \\ z_1 &= -z_2^2 + z_2^2 z_3 \\ z_2 &= -z_3^2 + z_3^2 z_4 \\ z_3 &= -z_4^2 + z_4^2 z_5 \\ z_4 &= -z_5^2 + z_5^2 z_6 \\ z_5 &= -z_6^2 - 22z_6^3 + O(z_6^4) \end{aligned}$

Die Hamburger-Noether-Entwicklungen von p_1 bis p_6 sind in Tabelle 6.1 zusammengestellt. Es ist jeweils $z_0 = [x]$ und $z_{-1} = [y]$. Die HNE von p_7 wurde nur in Charakteristik 2 berechnet. Sie lautet

$$p_7: \qquad \begin{aligned} z_{-1} &= z_0\, z_1 \\ z_0 &= z_1^2 + z_1^2 z_2 \\ z_1 &= z_2^2 + z_2^2 z_3 \\ z_2 &= z_3^2 + z_3^2 z_4 \\ z_3 &= z_4^2 + z_4^2 z_5 \\ z_4 &= z_5^2 + z_5^2 z_6 \\ z_5 &= z_6^2 + z_6^2 z_7 \\ z_6 &= z_7^2 + O(z_7^4) \end{aligned}$$

Beim Test der Effizienz der Implementation war das Ziel, zu der gegebenen Kurve in möglichst kurzer Zeit die Invarianten berechnen zu können. Dazu wurde als Ring in Singular `ring r=`p`,(x,y),ls;` deklariert mit $p \in \{0, 2, 101, 32003\}$ – die kleinstmögliche Charakteristik, die größtmögliche, eine „mittelgroße" und Null. Die Wahl dieses Ringes ist bei vorgegebener Charakteristik optimal im Hinblick auf die Rechenzeit (näheres dazu im Teil 6.2.5: „Intelligenter Einsatz von (`ext-`)`develop`" – Tabelle 6.10 auf S. 198 zeigt exemplarisch die Unterschiede bei p_5); andere Variablennamen oder Ringordnung ändern am Ergebnis nichts (siehe auch Abschnitt A.1.2 auf S. 208f).

Desweiteren wurde die Berechnung mit `develop(`p_i`,-1);` durchgeführt, d. h. sie beschränkt sich auf ein Minimum dessen, was für die Invariantenberechnung nötig ist.[1] Solange die HNE nicht abbricht, kann man schließlich beliebig viel Rechenzeit investieren, um die Berechnung fortzusetzen und beispielsweise eine genauere Parametrisierung der Kurve zu ermöglichen.

Bei der Analyse wurde zunächst die gesamte Rechenzeit untersucht, sowie der verbrauchte Speicher, der am Ende der Prozedur von `kmemory()` gemeldet wird, einer Prozedur aus der Bibliothek `general.lib`. Der Löwenanteil des verbrauchten Speichers entfällt dabei auf das transformierte Polynom, das näher untersucht wurde in Bezug auf Anzahl der Monome und Gesamtzahl der Zeichen, aus denen es besteht (`size(string(f))`). Die Rechenzeit wurde aufgeschlüsselt nach dem Verbrauch für den Quadratfreiheitstest (das ist die Prozedur `squarefree`, die einmal am Anfang aufgerufen wird), für die Irreduzibilitätstests, die beiden Transformationen T_1 und T, und für die Berechnung des Newtonpolygons (mit `getnm`).

Die Irreduzibilitätstests setzen sich zusammen aus zwei verschiedenen Tests, die im allgemeinen mehrfach ausgeführt werden, nämlich

- Besteht das Newtonpolygon aus genau einer Seite? (vgl. Schritt 2 im Algorithmus `HNE-irreduzibel`) Diesen Test erledigt `testreducible`.
- Ist die quasihomogene Leitform von der Gestalt $c\left(y-\delta x^Q\right)^N$? (vgl. Schritt 3) Hier wird sowohl die Zeit zur Berechnung von c und δ als auch zum Vergleich von $c\left(y-\delta x^Q\right)^N$ mit der Leitform berücksichtigt.

Normalerweise fällt auch noch Rechenzeit an für die Abbildung des Ergebnisses zurück in den Ring, von dem aus `develop` aufgerufen wurde; wegen der günstigen Wahl des Ringes `r` findet diese Abbildung aber nicht statt. Die Testergebnisse finden sich in den nachfolgenden beiden Tabellen.

Wie unschwer zu erkennen ist, gewinnt die Transformation T_1 mit wachsender Komplexität des Problems die alles überragende Bedeutung. Weil bei jeder Anwendung von T_1 die Anzahl der Monome stark zunimmt, entfällt der bei weitem größte Teil der angegebenen Rechenzeiten (mehr als 90%) von T_1 bzw. T auf die jeweils letzte Transformation dieses Typs.

In endlichen Körpern wirken zwei Faktoren gegeneinander: Einerseits profitieren wir davon, daß bei niedriger Charakteristik viele Monome wegfallen, was die Berechnung stark beschleunigen kann; andererseits ist das Potenzieren von Polynomen im Singular-Kern für kleine Charakteristiken nicht so schnell wie für große, was T_1 verlangsamt. Das führt zu dem bemerkenswerten Effekt, daß die Rechenzeiten für mittlere Charakteristiken verhältnismäßig ungünstig ausfallen.

In Körpern der Charakteristik 0 kommen zusätzlich die nicht mehr beschränkten Koeffizienten[2] hinzu. Das ist aber bei diesen Beispielen in erster Linie ein Speicher-

[1] Die in Tabelle 6.1 dargestellten HNEs sind nicht minimal; `develop(`p_i`,-1)`, $i>1$, liefert als letzte Zeile $z_{i-1}=-z_i^2+O(z_i^3)$

[2] In Charakteristik 2 bestehen alle Koeffizienten aus null Zeichen. Unter der Annahme, daß die Monome mit geraden und ungeraden Koeffizienten gleichmäßig verteilt sind, liefert die Spalte 2 der Tabelle der durchschnittlichen Zeichenzahlen gerade die durchschnittliche Anzahl der sonstigen Zeichen eines Monoms (im wesentlichen die Exponenten). In der Differenz ergibt sich, daß die Zahl der Zeichen für die Koeffizienten tatsächlich den zu erwartenden Werten von 1.8 für Charakteristik 101 und 4.31 für Charakteristik 32003 nahekommt.

Tabelle 6.2: Testergebnisse der Polynome von Merle

	Charakteristik							
	2		101		32003		0	
$\mathbf{p}_3$								
Speicher (kByte)	176		172		301		300	
Gesamtrechenzeit	0.16 sec		0.14 sec		0.2 sec		0.3 sec	
transform. Polynom								
Anzahl Monome	11		43		43		43	
Anzahl Zeichen	40		232		232		232	
$\mathbf{p}_4$								
Speicher (kByte)	197		207		336		342	
Gesamtrechenzeit	0.25 sec		0.44 sec		0.46 sec		0.56 sec	
davon für								
Quadratfreiheit	< 0.02 sec		0.08 sec	18%	0.09 sec	20%	0.16 sec	29%
Irreduzibilitätstests	0.04 sec	16%	0.08 sec	18%	0.05 sec	11%	0.05 sec	9%
Transformation T_1	< 0.02 sec		0.14 sec	31%	0.14 sec	30%	0.15 sec	27%
Transformation T	< 0.02 sec		0.04 sec	9%	0.02 sec	4%	< 0.02 sec	
Newtonpolygon	< 0.02 sec		< 0.02 sec		0.02 sec	4%	0.02 sec	4%
transform. Polynom								
Anzahl Monome	107		980		985		985	
Anzahl Zeichen	583		7596		9957		13 387	
$\mathbf{p}_5$								
Speicher (kByte)	244		680		1070		2237	
Gesamtrechenzeit	2.4 sec		87.2 sec		53.8 sec		78.5 sec	
davon für								
Quadratfreiheit	< 0.02 sec		0.9 sec	1.0%	0.9 sec	2 %	0.9 sec	1.1%
Irreduzibilitätstests	0.1 sec	4%	0.3 sec	0.3%	0.3 sec	0.6%	0.3 sec	0.4%
Transformation T_1	2.1 sec	87%	84.0 sec	96.3%	50.5 sec	94 %	74.3 sec	94.6%
Transformation T	0.04 sec	2%	1.7 sec	1.9%	1.8 sec	3 %	1.9 sec	2.4%
Newtonpolygon	< 0.02 sec		0.2 sec	0.2%	0.2 sec	0.4%	0.7 sec	0.9%
transform. Polynom								
Anzahl Monome	1887		18 445		22 424		22 425	
Anzahl Zeichen	13 627		165 882		260 468		991 236	
$\mathbf{p}_6$								
Speicher (kByte)	1028		11 066		13 114		81 150	
Gesamtrechenzeit	14.2 min		14 h 56 min		10 h 54 min		12 h 32 min	
davon für								
Quadratfreiheit	< 0.05 sec	0 %	11 sec	0.02%	12 sec	0.03%	12 sec	0.03%
Irreduzibilitätstests	0.3 sec	0.04%	8 sec	0.01%	12 sec	0.03%	20 sec	0.04%
Transformation T_1	14.1 min	99.1%	14.67 h	98.2%	10.48 h	96.1%	12.08 h	96.3%
Transformation T	7.0 sec	0.8%	15.4 min	1.7%	24.6 min	3.8%	26.3 min	3.5%
Newtonpolygon	0.5 sec	0.1%	6 sec	0.01%	7 sec	0.02%	24 sec	0.05%
transform. Polynom								
Anzahl Monome	39 581		507 994		561 935		561 957	
Anzahl Zeichen	338 358		5 328 641		7 311 885		108 246 986	

	Charakteristik 2	
$\mathbf{p_7}$		
Speicher (kByte)	21677	
Gesamtrechenzeit	82 h 23 min	
davon für		
Quadratfreiheit	< 0.05 sec	0 %
Irreduzibilitätstests	21 sec	0.01%
Transformation T_1	81.10 h	98.4%
Transformation T	1.27 h	1.5%
Newtonpolygon	13 sec	0 %
transform. Polynom		
Anzahl Monome	1 042 839	
Anzahl Zeichen	10 452 511	

Durchschnittliche Anzahl der Zeichen pro Monom

	Charakteristik			
	2	101	32003	0
p_3	3.6	5.4	5.4	5.4
p_4	5.4	7.8	10.1	13.6
p_5	7.2	9.0	11.6	44.2
p_6	8.5	10.5	13.0	192.6
p_7	10.0			

problem – die Rechenzeit erhöht sich etwa bei p_6 gegenüber 32003 „nur" um knapp 17% (und ist damit immer noch kürzer als bei 101!), während der Speicherbedarf auf mehr als das sechsfache steigt.

6.1.2 Reduzible Beispiele

Wir kommen zum Test der Prozedur `reddevelop`. Als erstes bietet sich an, die Prozedur mit den gleichen Beispielen wie für `develop` zu testen und zu vergleichen. Echte reduzible Beispiele lassen sich sehr einfach generieren, indem man mehrere Polynome p_i miteinander multipliziert. Zu beachten ist, daß `reddevelop` einen geeigneten Ring selbst generiert und automatisch aufhört zu rechnen, sobald alle Zweige beim Aufblasen getrennt und die Hamburger-Noether-Entwicklungen bis zum Beginn der letzten Zeile bekannt sind. Das bedeutet, daß die zurückgegebenen HNEs bei irreduziblen Kurven automatisch minimal sind (die berechnete Information ist also die gleiche, die Rechenzeiten sind direkt mit denen von `develop(`p_i`,-1)` vergleichbar) und daß die Wahl des Ringes für die Tests keine Rolle spielt.

Den Quadratfreiheitstest am Anfang erledigt wie zuvor `squarefree`. Anstelle von Irreduzibilitätstests müssen wir bei `reddevelop` das charakteristische Polynom ϕ_Δ faktorisieren. Die Transformationen T und T_1 sind ersetzt durch eine verallgemeinerte Transformation T_2. Das Newtonpolygon hat im allgemeinen mehrere Seiten und wird daher mit der Prozedur `newtonpoly` berechnet. Die Angaben über die transformierten Polynome sind der Tabelle 6.2 für den irreduziblen Fall zu entnehmen und werden hier nicht wiederholt.

Im Vergleich stellt man fest, daß der Speicherverbrauch etwas höher geworden ist (dies dürfte daran liegen, daß das Ergebnis erst von `HN` an `reddevelop` weitergereicht und umformatiert werden muß – bei `develop` entfällt dieser Transfer). Die Gesamtzeiten steigen bei den schnellen Beispielen aufgrund des erhöhten Verwaltungsaufwandes leicht an, der aber später nicht mehr ins Gewicht fällt. Bei den Einzelzeiten ist klar, daß der Quadratfreiheitstest gleich schnell bleibt (es handelt

Tabelle 6.3: Ergebnisse von p_i mit `reddevelop`	**Charakteristik**					
	2		32003		0	
$\mathbf{p_3}$						
Speicher (kByte)	205		332		332	
Gesamtrechenzeit	0.42 sec		0.43 sec		0.49 sec	
Quadratfreiheit	0.03 sec	7%	0.03 sec	7%	0.10 sec	20%
Faktorisierung	0.06 sec	14%	0.07 sec	16%	0.06 sec	12%
Transformation T_2	0.08 sec	19%	0.07 sec	16%	0.07 sec	14%
Newtonpolygon	0.03 sec	7%	0.03 sec	7%	0.03 sec	6%
$\mathbf{p_4}$						
Speicher (kByte)	220		381		404	
Gesamtrechenzeit	0.5 sec		0.74 sec		0.82 sec	
Quadratfreiheit	0.02 sec	4%	0.10 sec	14%	0.16 sec	20%
Faktorisierung	0.08 sec	16%	0.08 sec	11%	0.07 sec	9%
Transformation T_2	0.10 sec	20%	0.22 sec	30%	0.25 sec	30%
Newtonpolygon	0.02 sec	4%	0.04 sec	5%	0.06 sec	7%
$\mathbf{p_1\, p_2\, p_3\, p_4}$						
Speicher (kByte)	303		510		668	
Gesamtrechenzeit	1.94 sec		3.95 sec		4.45 sec	
Quadratfreiheit	0.08 sec	4%	1.09 sec	28%	1.22 sec	27%
Faktorisierung	0.14 sec	7%	0.13 sec	3%	0.14 sec	3%
Transformation T_2	0.30 sec	15%	1.10 sec	28%	1.26 sec	28%
Newtonpolygon	0.10 sec	5%	0.11 sec	3%	0.12 sec	3%
$\mathbf{p_5}$						
Speicher (kByte)	335		1453		3662	
Gesamtrechenzeit	2.78 sec		54.4 sec		84.3 sec	
Quadratfreiheit	0.02 sec	1%	0.9 sec	2 %	1.0 sec	1.2%
Faktorisierung	0.08 sec	3%	0.2 sec	0.4%	0.4 sec	0.5%
Transformation T_2	2.28 sec	82%	52.2 sec	96 %	79.7 sec	94.5%
Newtonpolygon	0.06 sec	2%	0.3 sec	0.6%	0.8 sec	0.9%
$\mathbf{p_4\, p_5}$						
Speicher (kByte)	1079		3239		9411	
Gesamtrechenzeit	63.0 sec		326 sec		457 sec	
Quadratfreiheit	0.03 sec	0.05%	4 sec	1.2%	4 sec	0.9%
Faktorisierung	0.2 sec	0.3%	0.3 sec	0.1%	0.8 sec	0.2%
Transformation T_2	60.9 sec	96.7%	318 sec	97.5%	439 sec	96.1%
Newtonpolygon	0.3 sec	0.4%	0.7 sec	0.2%	2 sec	0.4%
$\mathbf{p_6}$						
Speicher (kByte)	1831		23 361		155 863	
Gesamtrechenzeit	14.3 min		10 h 31 min		11 h 55 min	
Quadratfreiheit	0.02 sec	0 %	11 sec	0.03%	11 sec	0.03%
Faktorisierung	0.3 sec	0.03%	2 sec	0.01%	9 sec	0.02%
Transformation T_2	14.3 min	99.7%	10.5 h	99.9 %	11.88 h	99.7%
Newtonpolygon	0.5 sec	0.06%	7 sec	0.02%	23 sec	0.05%

sich um dieselbe Prozedur). Die Newtonpolygonberechnung mit `newtonpoly` ist nur geringfügig langsamer und trägt nach wie vor (ebenso wie der Quadratfreiheitstest) praktisch nichts zur Rechenzeit bei. Die Faktorisierung des charakteristischen Polynoms ist deswegen schneller als die Irreduzibilitätstests von `develop`, weil im unverzweigten Teil von `HN` (siehe Seite 161) die einzige Nullstelle von ϕ_Δ genauso wie im irreduziblen Fall berechnet wird, der Test, ob das Newtonpolygon aus einer Seite besteht, aber entfällt. Für kleine Beispiele macht sich bei T_2 bemerkbar, daß das Monom $x^s y^t$, durch das dividiert werden muß, bei reduziblen Polynomen erst a posteriori berechnet wird. Auch das spielt bei p_5 und p_6 keine Rolle mehr. In der Tabelle nicht aufgeführt wurde die Zeit, die Transfer und Umformatierung des Ergebnisses zwischen `HN` und `reddevelop` brauchen – diese wird erst bei $p_4\,p_5$ und p_6 meßbar und liegt relativ zur Gesamtzeit immer im Promillebereich (gleiche Größenordnung wie das Newtonpolygon).

Beim Test der echt reduziblen Beispiele $p_1\,p_2\,p_3\,p_4$ und $p_4\,p_5$ fällt auf, daß der Aufwand für den Quadratfreiheitstest besonders stark zunimmt; bei deutlich niedrigerer Gesamtrechenzeit liegt er im ersten Beispiel sogar absolut über dem für p_5. Dieser Trend wird sich bei den folgenden reduziblen Beispielen fortsetzen. Die Faktorisierung und die Newtonpolygonberechnung brauchen dagegen kaum mehr Zeit, als wenn man die HNE für die einzelnen Polynome getrennt ausrechnen würde.

Bei Robert J. Walker [Wa, III.2] finden sich die folgenden Beispiele:

$$
\begin{array}{ll}
Wa_1 = x^3 - x^2 + y^2 & Wa_6 = \left(x^2+y^2\right)^2 + 3x^2y - y^3 \\
Wa_2 = x^3 + x^2 + y^2 & Wa_7 = \left(x^2+y^2\right)^3 - 4x^2y^2 \\
Wa_3 = x^3 - y^2 & Wa_8 = x^6 - x^2y^3 - y^5 \\
Wa_4 = 2x^4 - 3x^2y + y^2 - 2y^3 + y^4 & Wa_9 = -x^3 + x^4 - 2x^2y - xy^2 + 2xy^4 + y^5 \\
Wa_5 = x^4 + x^2y^2 - 2x^2y - xy^2 + y^2 &
\end{array}
$$

Ihre Eigenschaften sind:

Wa_1	zwei glatte Zweige, transversaler Schnitt
Wa_2	zwei glatte Zweige, transversaler Schnitt (*)
Wa_3	irreduzibel, Puiseux-Paar (3,2) (siehe p_1)
Wa_4	zwei glatte Zweige mit Schnittmultiplizität 2
Wa_5	irreduzibel, Puiseux-Paar (5,2)
Wa_6	drei glatte Zweige, paarweise transversaler Schnitt (*)
Wa_7	vier glatte Zweige; jeder hat 2× Schnittmult. 1 und 1× Schnittmult. 2 mit den anderen drei Zweigen
Wa_8	drei transversale Zweige, davon zwei glatt und einer mit Puiseux-Paar (4,3) (*)
Wa_9	ein glatter Zweig und einer mit Puiseux-Paar (3,2), transversaler Schnitt

Bei den mit (*) gekennzeichneten Beispielen ist ein Ringwechsel nötig; das Ergebnis wurde in Charakteristik 32003 berechnet. Alle anderen Ergebnisse beziehen sich auf Charakteristik 0.

Tabelle 6.4: Beispiele von Walker

	Charakteristik 0				Charakteristik 32003			
	$Wa_{1,3,4,5}$		$Wa_{5,7,9}$		$Wa_{2,6,8}$		$Wa_{1,2,\ldots,9}$	
Speicher (kByte)	392		446		474		1754	
Gesamtrechenzeit	3.84 sec		7.19 sec		3.21 sec		16 min 9 sec	
davon für								
Quadratfreiheit	2.10 sec	55%	4.82 sec	67%	0.58 sec	18%	15.8 min	97.8%
Faktorisierung	0.16 sec	4%	0.22 sec	3%	0.35 sec	11%	1.1 sec	0.1%
Ringwechsel	—	0%	—	0%	0.04 sec	1%	0.2 sec	0.02%
Transformation T_2	0.14 sec	4%	0.27 sec	4%	0.17 sec	5%	9.7 sec	1.0%
Newtonpolygon	0.18 sec	5%	0.27 sec	4%	0.22 sec	7%	1.6 sec	0.2%
Weiterverarbeitung	0.15 sec	4%	0.23 sec	3%	0.42 sec	13%	3.0 sec	0.3%

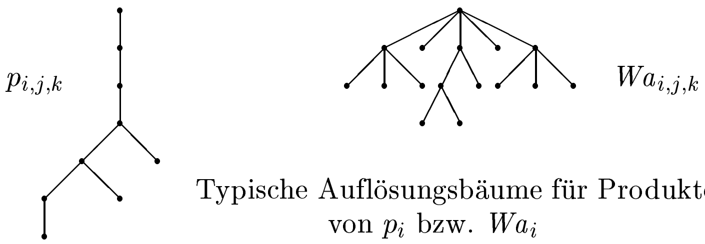
	Charakteristik 0						Char 32003		
	Wa_1	Wa_3	Wa_4	Wa_5	Wa_7	Wa_9	Wa_2	Wa_6	Wa_8
Speicher (kB)	306	308	316	318	337	340	326	335	346
Zeit (sec)	0.59	0.21	0.57	0.31	1.17	0.43	0.73	0.73	0.87

Weil diese Beispiele so kurz sind, wurde hier auf eine Aufspaltung in die Einzelzeiten verzichtet. Natürlich lassen sich wieder mehrere Polynome zu größeren Beispielen zusammensetzen, wie das für Tabelle 6.4 getan wurde.

Bei der Zeit für den Ringwechsel wurde die Berechnung eines geeigneten Minimalpolynoms, die Definition des neuen Ringes und das Abbilden der Daten in diesen berücksichtigt. Bei den früheren Beispielen gab es keinen Ringwechsel, weil die gegebenen Polynome bereits über dem Grundkörper in p_i zerfielen, welche irreduzible Kurven definieren.

In der Zeile „Weiterverarbeitung“ wurde der Aufwand zusammengefaßt, mit dem das Ergebnis von `HN` von der aufrufenden Prozedur weiterverarbeitet wird. Die aufrufende Prozedur kann wieder `HN` sein; in diesem Fall wird die Zeit erfaßt, um den Anfang der HNE, der vom aufrufenden `HN` berechnet wurde, zusammenzusetzen mit der/den Fortsetzung(en), die das aufgerufene `HN` liefert (zweiter Teil von `HN`, Seite 161: „setze aus L' und L'' eine Entwicklungsliste H zusammen“). Oder `HN` wurde von `develop` aufgerufen; dann muß das Ergebnis der Prozedur in eine Liste von HNEs in dem auf Seite 157 beschriebenen Format konvertiert werden. Die Zeit, die `extractHNEs` dafür benötigt, wird ebenfalls berücksichtigt.

Sofort fällt auf, daß T_2 seine bei den p_i herausragende Rolle eingebüßt hat. Dies ist darauf zurückzuführen, daß nur wenige Transformationen *hintereinander* nötig sind; viele Zweige schneiden sich transversal, werden daher auf Anhieb voneinander getrennt, und sind glatt. Beides führt dazu, daß es zwar viele Zweige gibt, die HNE jedes einzelnen Zweiges aber durch wenige Transformationen berechnet werden kann. Das hält das Wachstum der Polynome niedrig (auch zu erkennen am geringen Speicherverbrauch). Die Anzahl der Monome und die Größe ihrer Exponenten sind aber die wesentlichen Faktoren für den Aufwand von T_2.



Typische Auflösungsbäume für Produkte von p_i bzw. Wa_i

Demgegenüber wächst der Zeitbedarf für den Test auf Quadratfreiheit drastisch mit der Anzahl der Faktoren des gegebenen Polynoms (die Berechnung des ggT wird schwieriger). Daß die Faktorisierung in Ringen mit Minimalpolynom aufwendiger ist, könnte sich in Ansätzen bei den Beispielen mit Ringwechsel bemerkbar machen. Natürlich praktisch nur dort, wo das charakteristische Polynom mehrere Nullstellen hat, da sonst `factorize` umgangen wird.

Die nächsten Beispiele habe ich der Diplomarbeit [Fi] von Jochen Fink entnommen; sie stammen ursprünglich von V. Baladi. Der zugrundeliegende Körper ist $\mathbb{Q}$.

$$\begin{aligned}
Ba_1 &= x^{30} - 10y^2x^{27} + 45y^4x^{24} - 120y^6x^{21} - y^7x^{20} + 210y^8x^{18} + 8y^9x^{17} - 252y^{10}x^{15} - 25y^{11}x^{14} + \\
&\quad 210y^{12}x^{12} + 40y^{13}x^{11} - 120y^{14}x^9 - 35y^{15}x^8 + 45y^{16}x^6 + 16y^{17}x^5 + y^{18}x^4 - 10y^{18}x^3 - 3y^{19}x^2 + y^{20} \\
Ba_2 &= x^{14} - 3y^2x^{11} - y^3x^{10} - y^2x^9 + 3y^4x^8 + y^5x^7 + 3y^4x^6 + x^5(-y^6 + y^5) - 3y^6x^3 - y^7x^2 + y^8 \\
Ba_3 &= x^{11} - 2y^2x^8 - y^3x^7 - y^2x^6 + y^4x^5 + 2y^4x^3 + y^5x^2 - y^6 \\
Ba_4 &= x^{14} - 17x^{13} + x^{12}(12y + 16) - 72yx^{11} + 42y^2x^{10} - 40y^2x^9 + 48y^3x^8 - 2y^4x^7 + 33y^4x^6 - \\
&\quad 12y^5x^5 - 10y^6x^3 + y^8 \\
Ba_5 &= 100yx^{24} + x^{24} + 2y^{10}x^{12} + y^{20} \\
Ba_6 &= y^{12} + (y^{11} + 2y^9)x + 3x^2y^6 + 2x^3y^3 + x^4 \\
Ba_7 &= \big(y^{12} - 3x^2y^{11} - 6x^3y^{10} - x^3y^{11} + 3x^4y^{10} + 12x^5y^9 + 15x^6y^8 + 3x^6y^9 - 6x^7y^8 - 18x^8y^7 - \\
&\quad 20x^9y^6 - 6x^9y^7 + 3x^{10}y^6 + 12x^{11}y^5 + 15x^{12}y^4 + 4x^{12}y^5 - 3x^{14}y^3 - 6x^{15}y^2 - x^{15}y^3 + x^{18}\big) \\
&\quad \big(y^{12} - 3x^2y^{11} - 6x^3y^{10} + 2x^4y^{10} + 12x^5y^9 + 15x^6y^8 - x^6y^9 - 2x^7y^8 - 18x^8y^7 - 20x^9y^6 - \\
&\quad 3x^{10}y^6 + 12x^{11}y^5 + 15x^{12}y^4 + 4x^{13}y^4 - 3x^{14}y^3 - 6x^{15}y^2 - x^{16}y^2 + x^{18}\big) \\
Ba_8 &= Ba_4\,\big(y^4 - 18x^3y^2 - 12x^5y + 81x^6 - x^7\big) \\
Ba_9 &= xy\big(-y^{12} + 3x^2y^{11} + 6x^3y^{10} - 2x^4y^{10} - 12x^5y^9 - 15x^6y^8 + x^6y^9 + 2x^7y^8 + 18x^8y^7 + 20x^9y^6 + \\
&\quad 3x^{10}y^6 - 12x^{11}y^5 - 15x^{12}y^4 - 4x^{13}y^4 + 3x^{14}y^3 + 6x^{15}y^2 + x^{16}y^2 - x^{18}\big) \\
&\quad \big(y^4 - x^2y^3 - 2x^3y^2 + x^6\big)\,\big(x^3 - y^2\big)
\end{aligned}$$

Ihre Eigenschaften sind die folgenden:

Ba_1	irreduzibel, Puiseux-Paare $(3,2),(16,5),(33,2)$
Ba_2	drei Zweige, Puiseux-Paare: 1. $(3,2)$, 2. $(3,2),(7,2)$, 3. $(5,2)$, Schnittmultiplizitäten 6, 12 und 13
Ba_3	Zweige Nr. 2 und 3 von Ba_2 mit Schnittmultiplizität 12
Ba_4	zwei äquiresoluble Zweige mit Puiseux-Paaren $(3,2),(7,2)$ und Schnittmultiplizität 24
Ba_5	vier äquiresoluble Zweige mit Puiseux-Paar $(6,5)$; jeder Zweig hat $2\times$ die Schnittmultiplizität 30 und $1\times$ die 34
Ba_6	vier glatte Zweige mit je $2\times$ Schnittmultiplizität 3 und $1\times$ 4

Tabelle 6.5: Hamburger-Noether-Entwicklungen von einigen Ba_i

$$Ba_2: \quad \begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = z_1^2 \end{array} \qquad \begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = z_1^2 + z_1^2 z_2 \\ z_1 = z_2^2 - z_2^3 + z_2^4 + O\left(z_2^5\right) \end{array} \qquad \begin{array}{l} z_{-1} = z_0^2 z_1 \\ z_0 = z_1^2 \end{array}$$

$$Ba_5: \quad \begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = \frac{1}{25} a^2 z_1^5 \pm a\, z_1^8 + 50 z_1^{11} \mp \frac{39}{8} a^3 z_1^{14} + O\left(z_1^{17}\right) \\ z_{-1} = z_0 z_1 \\ z_0 = -\frac{1}{25} a^2 z_1^5 \pm \frac{1}{25} a^3 z_1^8 + 50 z_1^{11} \mp \frac{975}{8} a\, z_1^{14} + O\left(z_1^{17}\right) \end{array} \quad \text{mit} \quad a^4 = -625$$

$$Ba_9: \qquad [y] = z_{-1} = 0 \qquad [x] = z_{-1} = 0$$

$$\begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = z_1^2 \end{array} \qquad \begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = z_1^2 + z_1^2 z_2 \\ z_1 = z_2^2 - z_2^3 + z_2^4 + O\left(z_2^5\right) \end{array} \qquad \begin{array}{l} z_{-1} = z_0 z_1 \\ z_0 = z_1^2 + z_1^2 z_2 \\ z_1 = z_2^2 + z_2^2 z_3 \\ z_2 = z_3 z_4 \\ z_3 = -z_4^2 + 3 z_4^3 - 4 z_4^4 + O\left(z_4^5\right) \end{array}$$

Bei allen Zweigen ist, wo nicht anders angegeben, $[x] = z_0$ und $[y] = z_{-1}$.

Ba_7 zwei Zweige, Puiseux-Paare $(3,2),(7,2),(23,3)$ und $(3,2),(7,2),(25,3)$, Schnittmultiplizität 240

Ba_8 drei äquiresoluble Zweige mit Puiseux-Paaren $(3,2),(7,2)$ und paarweise Schnittmultiplizität 24

Ba_9 fünf Zweige; zwei sind glatt, die anderen haben die Puiseux-Paare $(3,2)$, $(3,2),(7,2)$ und $(3,2),(7,2),(23,3)$. Schnittmultiplizitäten: $1,2,3,4,6,12,13,18,39,80$

Die Polynome Ba_5 und Ba_6 erfordern für ihre Entwicklung eine Körpererweiterung. Da die Faktorisierung über Erweiterungen von $\mathbb{Q}$ in Singular noch nicht implementiert ist, wurde eine Spezialroutine geschrieben, die alle bei Ba_5 auftretenden Polynome korrekt faktorisieren kann. Damit ist es möglich, dieses Beispiel mit $\mathbb{Q}$ als Startkörper zu berechnen. Der Algorithmus benötigt zwei sukzessive Körpererweiterungen und wählt schließlich $a^4 + 625$ als Minimalpolynom, d.h. die HNE existiert über $\mathbb{Q}\left(5\sqrt[4]{-1}\right) = \mathbb{Q}\left(\sqrt[4]{-1}\right)$. Als Startkörper wurde auch $\mathbb{F}_{32003}$ versucht mit dem gleichen[3] Ergebnis wie oben; der einzige Unterschied besteht darin, daß dann die erste Körpererweiterung genügt; $x^4 + 625$ zerfällt bereits über $\mathbb{F}_{32003}[a] \,/\, \langle a^2+1 \rangle$ vollständig.

Ba_6 wurde nur mit $\mathbb{F}_{32003}$ als Startkörper berechnet. Auch hier stellte sich heraus, daß in Charakteristik 32003 eine Körpererweiterung genügt, wo in Charakteristik 0 mehrere gebraucht würden.

[3] d.h. die gefundenen Zweige haben jeweils das gleiche komplexe Modell; darüberhinaus stimmen alle Koeffizienten, die nicht vom Parameter a abhängen, in der HNE überein

Tabelle 6.6: Beispiele von Baladi

	Ba_5		Ba_6		Ba_7		Ba_8		Ba_9	
Gesamtzeit (sec)	1.87		1.64		9.63		1.52		2.14	
davon für										
Quadratfreiheit	0.04	2%	0.05	3%	8.65	90%	0.29	19%	0.97	45%
Faktorisierung	0.23	12%	0.26	16%	0.09	1%	0.13	9%	0.10	5%
Ringwechsel	0.10	5%	0.04	2%	—	0%	—	0%	—	0%
Transformation T_2	0.14	7%	0.03	2%	0.20	2%	0.23	15%	0.17	8%
Newtonpolygon	0.10	5%	0.13	8%	0.04	<1%	0.06	4%	0.05	2%
Weiterverarbeitung	0.22	12%	0.14	9%	0.08	1%	0.12	8%	0.18	8%

	Ba_1	Ba_2	Ba_3	Ba_4	Ba_5	Ba_6	Ba_7	Ba_8	Ba_9
Speicher (kB)	374	392	422	454	492	491	517	541	516
Zeit (sec)	0.83	0.83	0.69	0.92	1.87	1.64	9.63	1.52	2.14

Die Verteilung der Zeit auf die einzelnen Schritte findet sich für die Beispiele, die länger als eine Sekunde gebraucht haben, in Tabelle 6.6.

Zum Schluß untersuchen wir noch vier eigene Beispiele. Auf der Suche nach Beispielen mit einer „interessanten" Hamburger-Noether-Entwicklung (viele Zeilen, unterschiedliche Koeffizienten) bin ich den umgekehrten Weg gegangen: Man gebe sich eine Hamburger-Noether-Entwicklung vor und berechne durch Rückwärtseinsetzen eine Parametrisierung $(x(t), y(t))$; dann erhält man durch Elimination von t aus dem Ideal $\langle x - x(t), y - y(t)\rangle \subset k[x, y, t]$ eine Gleichung für eine Kurve mit dieser Parametrisierung. Auf diese Weise kann man auch gleich testen, ob die berechnete HNE mit der vorgegebenen übereinstimmt.

Es stellt sich heraus, daß eine derart erzeugte Kurve normalerweise mehrere Zweige hat, darunter auch den mit der vorgegebenen Parametrisierung. Und zwar tritt dieses Phänomen immer dann auf, wenn $x(t)$ und $y(t)$ noch weitere gemeinsame Nullstellen besitzen außer $t = 0$.[4] Wegen der speziellen Form der Parametrisierung, die durch die erste Zeile der HNE vorgegeben ist ($y(t) = \sum_i a_{0i}\, x(t)^i + x(t)^{h_0} z(t)$ für ein geeignetes $z(t) \in k[[t]]$, oder umgekehrt), sind diese auch zu erwarten.

Das erste Beispiel entstand aus der Vorgabe von

$$\begin{aligned} z_{-1} &= z_0^3 + z_0^3 z_1 \\ z_0 &= z_1^3 z_2 \\ z_1 &= 1z_2^2 + 2z_2^3 + z_2^3 z_3 \\ z_2 &= 3z_3^2 + 4z_3^3 \end{aligned} \quad \widehat{=} \quad \text{HN-Matrix} \quad \begin{pmatrix} 0 & 0 & 1 & x \\ 0 & 0 & 0 & x \\ 0 & 1 & 2 & x \\ 0 & 3 & 4 & x \end{pmatrix}$$

als HNE. Das Ergebnis ist ein Polynom $La_1 \in \mathbb{F}_{32003}[x, y]$ bestehend aus 158 Monomen (wir verzichten hier auf eine Wiedergabe[5]), welches eine Kurve mit sechs

[4] $\langle x - x(t), y - y(t)\rangle \cap k[x, y]$ definiert die Projektion einer Raumkurve auf die (x, y)-Ebene; für jede Umgebung einer gemeinsamen Nullstelle von $x(t)$ und $y(t)$ erhalten wir dabei einen Zweig.

[5] Zur Konstruktion verwende man z. B. folgende Singular-Kommandos:

Zweigen definiert. Zu ihrer Berechnung ist ein Wechsel in den Körper mit Minimalpolynom $a^2 - 709a - 6349$ erforderlich. Der vorgegebene Zweig hat die Puiseux-Paare $(23, 7), (49, 2)$ und schneidet von den übrigen Zweigen vier mit Multiplizität 138 (Puiseux-Paar jeweils $(10, 3)$) und einen mit Multiplizität 324 (Puiseux-Paar $(23, 7)$).

Um $La_2 \in \mathbb{F}_7[x, y]$ zu erzeugen, wurde die HNE

$$\begin{aligned} z_{-1} &= 2z_0^2 + z_0^2 z_1 \\ z_0 &= z_1^3 z_2 \\ z_1 &= 3z_2^2 + 2z_2^5 + z_2^6 z_3 \\ z_2 &= z_3^2 + 2z_3^3 \end{aligned}$$

vorgegeben.[6] La_2 ist sogar noch größer: Es besteht aus 283 Monomen. Die dadurch gegebene Kurve besteht aus 15 Zweigen und erfordert eine Körpererweiterung mit Minimalpolynom $a^{11} + a^{10} + 3a^9 - a^5 - 2a^4 + 3a^3 - 2a^2 - a + 2$. 13 Zweige haben das Puiseux-Paar $(7, 3)$, einer hat $(16, 7)$ und der vorgegebene Zweig hat die beiden Paare $(16, 7), (41, 2)$. Seine Schnittmultiplizität mit den anderen Zweigen ist einmal 232 (Trennung nach $11\times$ Aufblasen) und sonst 96 (Trennung nach 5 quadratischen Transformationen).

Das folgende Polynom besitzt eine hohe Anzahl ähnlicher Hamburger-Noether-Entwicklungen; bei jeder quadratischen Transformation außer der ersten werden Zweige abgespalten – das führt zu einer hohen Rekursionstiefe bei der Berechnung.

$$\begin{aligned} La_3 = {} & \left(x^{11} - 2x^{10} + 2x^8 y - x^8 + 4x^6 y - 6x^4 y^2 + 4x^2 y^3 - y^4\right) \cdot \left(4x^{11} - 2x^{10} + 2x^8 y - x^8 + 4x^6 y - \right. \\ & \left. 6x^4 y^2 + 4x^2 y^3 - y^4\right) \cdot \left(3x^{14} - x^{13} + 2x^{12} + x^{11} y - 4x^{10} y + x^{10} + 2x^8 y^2 - 5x^8 y + 10x^6 y^2 - \right. \\ & \left. 10x^4 y^3 + 5x^2 y^4 - y^5\right) \cdot \left(2x^{12} - 4x^{11} + 3x^{10} + 8x^9 y - 13x^8 y - 4x^7 y^2 + 22x^6 y^2 - 18x^4 y^3 + \right. \\ & \left. 7x^2 y^4 - y^5\right) \cdot \left(x^5 - x^4 + 2x^2 y - y^2\right) \cdot \left(2x^8 + x^6 - 3x^4 y + 3x^2 y^2 - y^3\right) \cdot \left(x^7 - 2x^6 - 4x^5 - 2x^4 + \right. \\ & \left. 4x^3 y + 4x^2 y - 2y^2\right) \cdot \left(x^{25} + x^{24} - 4x^{23} - x^{22} y + 4x^{22} + 8x^{21} y - 2x^{21} - 12x^{20} y - 4x^{19} y^2 + \right. \\ & 4x^{20} + 10x^{19} y + 12x^{18} y^2 - 24x^{18} y - 20x^{17} y^2 - 4x^{16} y^3 + x^{18} + 60x^{16} y^2 + 20x^{15} y^3 - 9x^{16} y - \\ & 80x^{14} y^3 - 10x^{13} y^4 + 36x^{14} y^2 + 60x^{12} y^4 + 2x^{11} y^5 - 84x^{12} y^3 - 24x^{10} y^5 + 126x^{10} y^4 + 4x^8 y^6 - \\ & \left. 126x^8 y^5 + 84x^6 y^6 - 36x^4 y^7 + 9x^2 y^8 - y^9\right) \cdot \left(x^7 - 2x^4 + 4x^2 y - 2y^2\right) \in \mathbb{F}_{32003}[x, y] \end{aligned}$$

```
ring r=32003,(x,y,t),ds;
ideal i=-x+2187t14-11591t15-6998t16+12970t17-8629t18-3556t19-593t20-5630t21+15334t22+1752t23+1059t24+13462t25+8910t26
-2441t27-13350t28+7656t29-4450t30-1179t31+10450t32-7523t33,
-y+12638t42+1831t43+13467t44-101t45+9356t46+14018t47+4343t48+325t49+11514t50-15431t51-14076t52+3800t53-9820t54-14576t55
+15580t56+10729t57+13613t58+10437t59+15646t60-14518t61+9001t62-2467t63+13679t64+5106t65-10523t66-7818t67+9826t68
+1003t69+12137t70+11474t71+6875t72+2571t73+475t74+14677t75+5977t76+2540t77+14396t78-12742t79-12783t80-14280t81-2544t82
+6063t83+4785t84+14286t85-9139t86+2261t87+15907t88-7928t89+4629t90+15010t91+14410t92-8579t93+1226t94+3045t95-710t96
+7330t97+9214t98-14307t99+2757t100-1499t101+14045t102-7733t103+926t104-15303t105-5625t106+12681t107+13548t108-9173t109;
poly La1=eliminate(i,t)[1];
```

[6] Konstruktion: `ring R=7,(x,y,t),ds;`

```
ideal i=-x-t14-2t20-3t22-3t23-t24-3t25-3t26-t27+t28+t29+3t30+t31-2t32+2t33+3t34-t35-3t36+t37+3t38-t39+t40-t41+3t43-2t44
-2t45-2t46-t47-3t48+3t49-2t53+2t54+2t55-3t56-t57-2t58-2t59+2t60,
-y+2t28+3t32-2t33-t34-2t36-2t37-3t38+3t39-2t40+3t41-t42-t43-3t46-2t47-3t48-3t50-3t51+3t52-t53-t54-2t55-t56+t57+2t58+t59
+2t60-3t61-2t62-2t63-3t64-3t65+3t66+2t67+2t68-2t69+2t70-2t71-t72-2t73-2t74+2t75-3t76+2t77-3t78-3t79-3t80+t81-3t82-2t83
-2t84+2t85+t86-2t87-3t88+2t90-2t91+2t92+2t93-2t94+3t95+t96-3t97+3t99-3t100+t103-2t104-2t105-2t106-3t107-t109-2t110
-3t111+3t112-t113-2t114-t115-2t116-3t117+2t118-2t119+t120+2t123+t124+t125+3t130-2t131-2t132+t137-3t138-3t139;
poly La2=eliminate(i,t)[1];
```

Tabelle 6.7: Eigene Beispiele

	La_1		La_2		La_3		La_4	
gewählter Körper	$\mathbb{F}_{32003^2}$		$\mathbb{F}_{7^{11}}$		$\mathbb{F}_{32003^2}$		$\mathbb{F}_{32003^{63}}$	
Speicher (kByte)	504		587		1031		621	
Gesamtrechenzeit	4.11 sec		37.57 sec		440.6 sec		3 h 40 min	
davon für								
Quadratfreiheit	0.91 sec	22 %	6.30 sec	17 %	430.1 sec	97.6 %	$<$ 0.05 sec	0 %
Faktorisierung	0.29 sec	7 %	23.61 sec	63 %	0.5 sec	0.1 %	3.64 h	99.4 %
Ringwechsel	0.12 sec	3 %	0.26 sec	1 %	1.2 sec	0.3 %	2.5 sec	0.02%
Transformation T_2	1.20 sec	29 %	4.17 sec	11 %	4.4 sec	1.0 %	77.5 sec	0.6 %
Newtonpolygon	0.11 sec	3 %	0.26 sec	1 %	0.5 sec	0.1 %	$<$ 0.05 sec	0 %
Weiterverarbeitung	0.30 sec	7%	0.69 sec	2 %	1.1 sec	0.2 %	0.1 sec	0 %

Die zweite Zeile gibt den Körper an, den `reddevelop` zur Berechnung der HNE letztendlich wählt.

Die Multiplizitätensequenz der Kurve lautet

$$
\begin{array}{l}
[(1,2,4,3,3,3,6,3,1,1,1,1,1,1,1,2,2)],\\
[(1,2,4,3,3,3,6,3,1,1,1,1,1,1,1,2,2)],\\
[(1),(1,2,2,2,2,4,2,1,1,1,1,1,1,1,2,2)],\\
[(1),(1,2,1,1,1,2,1),(1),(1),(1),(1),(1),(1),(1),(1),(1)],\\
[(1),(1,1),(1,1,1,2,1),(1),(1),(1),(1),(1),(1),(1),(1),(1)],\\
[(1),(1),(1),(1,1,1,2,1),(1),(1),(1),(1),(1),(1),(1),(1),(1)],\\
[(1),(1),(1),(1),(1),(1,1,1),(1),(1),(1),(1),(1),(1),(1),(1),(1)],\\
[(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1),(1)],\\
\qquad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots
\end{array}
$$

Wie man sofort sieht, erhalten wir 17 Zweige, wovon acht glatt sind. Die übrigen Zweige haben folgende Puiseux-Paare:

$$1 \times (5,2), \quad 1 \times (5,2),(11,2), \quad 4 \times (8,3), \quad 1 \times (8,3),(19,2) \quad \text{und} \quad 2 \times (7,2)$$

Die gewählte Körpererweiterung hat das Minimalpolynom $a^2 - 9380a - 6469$. Die 136 Schnittmultiplizitäten brauchen wir nicht mehr angeben; der geneigte Leser kann sie mit Hilfe von Lemma 4.11 direkt aus der Multiplizitätensequenz berechnen.

Das vierte und letzte Beispiel sieht ganz einfach aus, hat es aber in sich. Es handelt sich dabei um ein homogenes Polynom mit zufällig gewählten Koeffizienten.

$$La_4 = 3043y^{18} + 18876x^4y^{14} + 6551x^9y^9 + 18906x^{17}y + 60x^{18} \in \mathbb{F}_{32003}[x,y]$$

Die resultierende Kurve besteht aus 18 glatten sich paarweise transversal schneidenden Zweigen und benötigt zu ihrer Berechnung zwei sukzessive Körpererweiterungen. Das zuletzt gewählte primitive Element hat ein dicht besetztes Minimalpolynom vom Grad 63.

In der Tabelle mit den Testergebnissen stellt man wiederum fest, daß T_2 keine große Rolle spielt, sondern teils der Quadratfreiheitstest, teils (und das ist neu) die Faktorisierung des charakteristischen Polynoms den größten Aufwand bereitet. Dabei zeigt sich, daß die Rechenzeit für die Faktorisierung entscheidend vom Grad der benötigten Körpererweiterung abhängt (= Grad des Minimalpolynoms). Alle Beispiele von Walker kommen mit Körpererweiterungen vom Grad 2 aus, falls sie überhaupt eine benötigen, bei den Beispielen von Baladi haben wir die Grade 2 und 4; aus diesem Grund stoßen wir erst jetzt auf Schwierigkeiten.

An dieser Stelle ist zu bemerken, daß das Beispiel La_4 schon sehr extrem ist: Ich habe es gezielt so konstruiert, daß ein charakteristisches Polynom vom Grad 18 herauskommt (ein so hoher Grad dürfte sonst eher selten sein; ohne solche Polynome erreicht man aber in Charakteristik > 0 nur schwer hochgradige Körpererweiterungen). Da die Zweige der Kurve glatt sind und sich transversal schneiden, beschränkt sich die Berechnung der HNE praktisch auf die Faktorisierung dieses Polynoms über seinem Zerfällungskörper; danach genügt einmal Aufblasen, um die Zweige[7] voneinander zu trennen, und wir sind fertig. Ein so hoher Anteil der Faktorisierung an der Gesamtrechenzeit sollte nach meiner Einschätzung also ein Ausnahmefall sein.

Noch eine kurze Erklärung zu den Zeiten für den Ringwechsel: Der vergleichsweise (zu Wa_i und Ba_i) hohe Zeitbedarf kommt bei La_1, La_2, La_3 vorwiegend durch die Abbildung der Daten in den neuen Ring zustande. Bei diesen drei Beispielen erfolgt der Ringwechsel erst relativ spät; einige HNE's sind bereits fertig berechnet und müssen ebenfalls in den neuen Ring transferiert werden. Die Zeit für den Ringwechsel selbst (Berechnung des Minimalpolynoms und Ringdefinition) beträgt (incl. Abb. des charakt. Polynoms) bei La_1 0.05 sec, bei La_2 0.04 sec, bei La_3 0.13 sec und liegt damit ganz im Bereich der bisherigen Beispiele. Nur bei La_4, wo der Ringwechsel ganz zu Beginn der Rechnung erfolgt, erfordert die Berechnung des Minimalpolynoms mit 2.45 sec fast die ganze Zeit.

6.1.3 Zusammenfassung

Die durchgeführten Tests weisen darauf hin, daß die Berechnung des Newtonpolygons, die Irreduzibilitätstests sowie der Ringwechsel generell eine völlig untergeordnete Rolle spielen. Rechenzeit und Speicherbedarf hängen dagegen stark ab von der Anzahl der Transformationen T_1 oder T_2, die hintereinander ausgeführt werden müssen (d. h. von der Anzahl der von Null verschiedenen Einträge in der Hamburger-Noether-Matrix, von der letzten (unendlichen) Zeile abgesehen). Die Transformation T ist im Aufwand gegenüber T_1 fast vernachlässigbar, deswegen ist es unerheblich, ob man mehrere T zusammenfaßt in T_2 oder nicht (in der Praxis treten ohnehin nur wenige T hintereinander auf – bei den Beispielen p_i etwa gibt es nirgends zwei aufeinanderfolgende T). Aus diesem Grund wurde die Prozedur

[7] weil es davon 18 gibt, kommt für die 18 Polynome $T_1(La_4, \delta_i, 1)$, die zu berechnen sind, immerhin noch über eine Minute zusammen

`develop` auch nicht mehr nachträglich von T & T_1 auf T_2 umgestellt. Da im irreduziblen Fall keine zwei Transformationen auf das gleiche Polynom angewandt werden, besteht bei umfangreichen Beispielen der größte Aufwand[8] stets in der Berechnung von T_1.

Im reduziblen Fall ist das Bild nicht mehr so einheitlich. Für Quadratfreiheitstest, Faktorisierung des charakteristischen Polynoms und Transformation T_2 gibt es jeweils Beispiele, bei denen mehr als 97 % der Zeit für einen einzigen dieser Posten verbraucht wird. Wenige Zweige bewirken normalerweise einen hohen Anteil von T_2. Bei Beispielen, die bereits über dem anfänglich gegebenen Polynomring in viele Faktoren zerfallen, überwiegt der Quadratfreiheitstest. In seltenen Fällen, wenn die erforderliche Körpererweiterung einen hohen Grad hat und komplizierte charakteristische Polynome auftreten, dominiert deren Faktorisierung.

Bei kurzen Beispielen (weniger als 2 Sekunden Gesamtrechenzeit) fallen noch die Verwaltung der Daten (insbesondere die Umformatierung und Zusammensetzung der HNEs) ins Gewicht sowie sonstige nicht mit Einzelzeit erfaßte Teile. Dazu gehört z. B. die Überprüfung des Ringes, der beim Aufruf aktiv ist (vgl. Tabelle A.1 auf Seite 213). Generell spielen diese Dinge bei längeren Beispielen keine Rolle mehr.

6.2 Verbesserung der Effizienz

Wir kommen nun zu der Frage, welche Verbesserungen im Laufe der Entwicklung der Bibliothek vorgenommen wurden, um die in 6.1 ermittelte Performance zu erhalten.

6.2.1 Quadratfreiheitstest

Wie schon bei der Beschreibung der Implementation von `develop` in Abschnitt 5.2.2 auf Seite 158f erwähnt, kann dieser Test für Polynome über $\mathbb{Q}$ auch in Charakteristik 32003 statt 0 durchgeführt werden. Theoretisch wäre das auf Körpererweiterungen übertragbar, ich habe jedoch aus drei Gründen darauf verzichtet: Erstens stellt sich bei algebraischen Körpererweiterungen die Frage, welcher Körper der Charakteristik 32003 überhaupt gewählt werden soll (definiert man ein Minimalpolynom, müßte man gewährleisten, daß es irreduzibel ist), zweitens ist eine Abbildung $\mathbb{Q}(a)[x, y] \to \mathbb{F}_{32003}(a)[x, y]$ in Singular überhaupt nicht implementiert, und drittens wird man normalerweise $\mathbb{Q}$ als Startkörper bevorzugen, so daß man nur sehr selten einen Nutzen davon hätte, würde man hier noch zusätzliche Arbeit investieren.

[8] Leider sind mir keine zeitaufwendige irreduzible Beispiele außer den p_i bekannt, um diese Behauptung zu überprüfen, aber die einzige denkbare Ausnahme wären Polynome, die sich schwer auf Quadratfreiheit überprüfen lassen und eine kurze HNE haben. Es hat aber den Anschein, daß sich die Anzahl der Faktoren eines Polynoms stärker auf den Quadratfreiheitstest auswirkt als die Größe.

Es kann natürlich passieren, daß diese Maßnahme die Rechenzeit verlängert, anstatt sie zu verkürzen

- wenn die Zeit für den Ringwechsel und die Abbildung des Polynoms in den neuen Ring länger dauert als die eingesparte Zeit: Da das aber sehr schnell geht, kann man selbst im ungünstigsten Fall nur Sekundenbruchteile[9] verlieren.
- wenn das tatsächlich quadratfreie Polynom über $\mathbb{F}_{32003}$ nicht quadratfrei ist und der Test über $\mathbb{Q}$ wiederholt werden muß: Das ist aber sehr unwahrscheinlich – sämtliche Beispiele aus Abschnitt 6.1 werden korrekt als quadratfrei erkannt.

Diese Nachteile zählen wenig gegenüber dem Zeitgewinn bei fast allen größeren Beispielen:

Zeiten in sec	p_5	p_6	$p_1\,p_2\,p_3\,p_4$	$p_4\,p_5$	Ba_7	La_3
Char 0	4.45	106.3	6.0	26.9	56.8	5989
Char 32003	0.84	11.6	1.1	3.9	8.5	442
Einsparung	81 %	89 %	82 %	86 %	85 %	92.6 %

6.2.2 Transformation T_1

$$T_1(f,\delta,Q)(x,y) := f\left(x, y+\delta x^Q\right)$$

Da diese Transformation für irreduzible Beispiele äußerst zeitkritisch ist, lohnt sich eine Investition an dieser Stelle besonders. Verantwortlich für die hohe Rechenzeit ist die Stelle, an der jedes Monom $x^\alpha y^\beta$ des zu transformierenden Polynoms ersetzt wird durch $x^\alpha\left(y+\delta x^Q\right)^\beta$. Beispielsweise hat das transformierte p_5 (wenn die letzte Zeile der HNE erreicht ist) bereits über 2000 Monome mit $\beta \geq 100$. Anstatt $y+\delta x^Q$ als allgemeines Polynom zu potenzieren, lohnt sich die Anwendung der binomischen Formel:

$$x^\alpha\left(y+\delta x^Q\right)^\beta = x^\alpha \sum_{i=0}^{\beta} \binom{\beta}{i} \delta^i\, x^{Qi} y^{\beta-i}$$

Diese wurde im Singular-Kern ab Version 0.9.3a implementiert. Es folgt ein Vergleich, wie lange die jeweils letzte Transformation T_1, die bei der Berechnung der HNE von p_4 bzw. p_5 durchgeführt wird, unter verschiedenen Singularversionen braucht. Die Prozentangaben beziehen sich auf die Zeit, die gegenüber Singular 0.9.2f eingespart werden konnte.

[9] Die zusätzlich benötigte Zeit läßt sich den Tabellen für die p_i entnehmen: In den Spalten für Charakteristik 32003 und 0 wurde jeweils das gleiche Polynom getestet, nur daß in Charakteristik 0 zusätzlich noch der Ringwechsel nötig ist: Der Unterschied macht etwa 0.07 Sekunden aus.

Zeit für T_1	Charakteristik			
	2	101	32003	0
$\mathbf{p_4}$				
Singular 0.9.2f		0.4 sec	0.4 sec	0.4 sec
Singular 0.9.3a		0.12 sec 70%	0.12 sec 70%	0.12 sec 70%
Singular 1.3.0		0.11 sec 73%	0.10 sec 75%	0.12 sec 70%
$\mathbf{p_5}$				
Singular 0.9.2f	2.1 sec	133 sec	160 sec	442 sec
Singular 0.9.3a	1.9 sec 10%	90 sec 32%	61 sec 62%	74 sec 83%
Singular 1.3.0	1.9 sec 10%	84 sec 37%	53 sec 67%	67 sec 85%

Vor dem Einsatz der binomischen Formel (mit Singular 0.9.2f) ergibt sich das zu erwartende Bild: Je mehr Elemente der Körper hat, desto langsamer die Berechnung (es fallen immer weniger Monome durch Koeffizienten, die zu Null werden, weg). Jedoch hängt die Geschwindigkeitssteigerung bei hohen auftretenden Exponenten ebenfalls stark von der Charakteristik ab, und zwar gerade umgekehrt. Bei p_4 ist noch kein Unterschied festzustellen (der größte Exponent β ist 25 – deutlich kleiner als die Charakteristiken, in denen die Berechnung durchgeführt wurde), aber bei p_5 (hier nimmt β Werte bis zu 125 an) nützt uns die Formel in Charakteristik 2 fast gar nichts und in Charakteristik 101 relativ wenig. Beim Vergleich 101 – 32003 – 0 bemerkt man jeweils eine ungefähre Halbierung im Verhältnis der Zeiten von 1.3.0 und 0.9.2f. Daß durch diesen Effekt die Gesamtrechenzeit von p_5 und p_6 in Charakteristik 101 höher ist als in 32003 und 0, haben wir ja schon in Abschnitt 6.1 gesehen. Der Einsatz der binomischen Formel bewirkt für Körper mit großer Charakteristik oder Null bei den irreduziblen Beispielen p_i die größte Steigerung der Rechengeschwindigkeit.

6.2.3 Transformation T_2

Diese Transformation profitiert natürlich genauso von der binomischen Formel im Singular-Kern, da sie die Transformation T_1 enthält. Zusätzlich ergibt sich aber bei reduziblen Polynomen die Schwierigkeit, s und t zu finden, s. d.

$$T_2(f, \delta', M', N') = T_1\left(\frac{T^*(f, M', N')}{x^s y^t}, \delta', Q\right)$$

Wie schon bei der Implementation von `T2_Transform` erklärt, sind s und t die gleichen wie bei der Transformation eines Referenzpolynoms, welches das gleiche Newtonpolygon wie f besitzt. Ein gutes Referenzpolynom ist ohne Aufwand zu ermitteln, weil das Newtonpolygon von f zum Zeitpunkt des Aufrufs von `T2_Transform` bereits bekannt ist.

Der Performancegewinn ist jedoch meistens nur gering, weil die Berechnung von s und t schnell ist im Vergleich zu T_1. Insbesondere läßt sich bei irreduziblen Polynomen die Berechnung von s (welches Null ist) kaum beschleunigen. Normalerweise spart man mit dieser Maßnahme ungefähr 5 % der Rechenzeit ein.

Zeit für T_2	p_4	$p_1\,p_2\,p_3\,p_4$	p_5	$p_4\,p_5$	p_6
ohne refpoly	0.28 sec	1.68 sec	78.5 sec	472 sec	12.8 h
mit refpoly	0.27 sec	1.51 sec	76.0 sec	450 sec	12.2 h
	$Wa_{1,2,\ldots,9}$	La_1	La_2	La_3	
ohne refpoly	15.2 sec	1.26 sec	5.11 sec	8.45 sec	
mit refpoly	11.1 sec	1.19 sec	4.16 sec	4.49 sec	

6.2.4 Newtonpolygon

Die Berechnung ging im irreduziblen Fall ursprünglich mit

```
proc getnm (poly f)
{ return( ord(f - (f/x)*x), ord(f - (f/y)*y) ); }
```

Dabei wird benutzt, daß Singular bei f/x die Monome durch Null ersetzt, bei denen die Division nicht aufgeht. Der `subst`-Befehl (siehe Listing von `getnm` in 5.2.1) erwies sich aber als deutlich schneller (die Prozentangaben rechts beziehen sich auf die Gesamtzeit):

Zeit für `getnm`	p_4	p_4 ext	p_5	p_5 ext
ohne `subst`	0.06 sec 10%	0.15 sec 10%	2.54 sec 3%	6.28 sec 0.9%
mit `subst`	0.02 sec 4%	0.06 sec 4%	0.80 sec 1%	1.65 sec 0.2%
Einsparung	67 % 7%	60 % 3%	69 % 2%	74 % 0.1%

Bei p_4 ext und p_5 ext handelt es sich um eine Fortsetzung der (minimalen) HNE von p_4 bzw. p_5; es wurde ein zusätzlicher Koeffizient der HN-Matrix berechnet. Die Prozentangaben auf der rechten Seite beziehen sich jeweils auf den Anteil an der Gesamtrechenzeit. Die Zeit für `getnm` reduziert sich also deutlich, allerdings relativiert sich die Einsparung im Vergleich zur Gesamtzeit, weil diese Prozedur nur einen geringen Anteil daran hat.

Ganz anders stellt sich das beim reduziblen Fall dar: Hier kam ursprünglich eine in C geschriebene Prozedur zum Einsatz, die in den Singular-Kern eingebunden war. Leider erwies sie sich als viel zu langsam, weil sie ursprünglich für die Berechnung der vollständigen konvexen Hülle des Newtondiagramms ($D(f)$ anstatt $D(f)+\mathbb{R}^2_+$) gedacht war. Ich habe daraufhin eine eigene Prozedur `newtonpoly` geschrieben, die wirklich nur das berechnet, woran wir interessiert sind, und die davon Gebrauch macht, daß das Newtonpolygon keine Halbgeraden enthält. War vorher der Zeitbedarf bei den p_i sogar noch deutlich höher als der für die Transformation T_2, so sank der Aufwand auf das jetzt in allen Beispielen völlig untergeordnete Niveau, und das, obwohl die eigene Prozedur erst von Singular interpretiert werden muß. Diese Maßnahme hat (wohlgemerkt bei `reddevelop`) den größten Performancegewinn gebracht, sogar der Speicherbedarf hat sich deutlich verringert (vergleiche die folgende Tabelle).

6.2.5 Intelligenter Einsatz von (`ext-`)`develop`

Weil die Rechenzeit für T_1 für jede weitere Transformation so stark zunimmt, bot sich an, die Möglichkeit zu schaffen, deren Anzahl für aufwendige Fälle auf ein

Tabelle 6.8: Newtonpolygonberechnung am Beispiel der p_i in Charakteristik 0

	p_4	$p_1\,p_2\,p_3\,p_4$	p_5	$p_4\,p_5$
$N(f)$ im Kern				
Speicher (kByte)	506	1008	4589	12 751
Gesamtrechenzeit	6 sec	32 sec	33.3 min	165.8 min
Quadratfreiheit	< 1 sec	1 sec 3%	5 sec 0.3%	28 sec 0.3%
Faktorisierung	< 1 sec	< 1 sec	< 1 sec	1 sec 0 %
Transformation T_2	< 1 sec	2 sec 6%	82 sec 4 %	8.3 min 5 %
Newtonpolygon	5 sec 83%	25 sec 78%	31.8 min 96 %	157.0 min 95 %
`newtonpoly`				
Speicher (kByte)	308	330	817	878
Gesamtrechenzeit	1 sec	8 sec	89 sec	8.8 min
Quadratfreiheit	< 1 sec	1 sec 13%	6 sec 7 %	28 sec 5 %
Faktorisierung	< 1 sec	< 1 sec	< 1 sec	1 sec 0.2%
Transformation T_2	< 1 sec	1 sec 13%	81 sec 91 %	8.1 min 93 %
Newtonpolygon	< 1 sec	< 1 sec	1 sec 1 %	3 sec 0.6%
Einsparung				
insgesamt	83 %	75 %	95.5 %	94.7 %
Newtonpolygon	> 80 %	> 96 %	99.95 %	99.97 %

Minimum zu reduzieren. Dies wurde durch den optionalen zweiten Parameter bei `develop` realisiert – ist er negativ, stoppt die Berechnung, sobald die letzte Zeile der HNE erreicht ist. Eine solche „minimale" Hamburger-Noether-Entwicklung erlaubt immer noch die Berechnung aller Kurveninvarianten, allerdings hat die HN-Matrix im allgemeinen undefinierte Einträge, was standardmäßig vermieden wird. Die Prozedur `extdevelop` erlaubt es, die HNE nachträglich fortzusetzen, ohne die bisherige Rechnung wiederholen zu müssen.

Bei `reddevelop` ist in dieser Hinsicht nichts zu tun: Bei reduziblen Kurven wird die normale Vorgehensweise die sein, zuerst die Anzahl der Zweige und ihre Invarianten zu ermitteln. Eventuell interessiert man sich dann für einen oder mehrere Zweige näher; von diesen kann man sich mit `extdevelop` die HNE verlängern lassen. Darum bricht `reddevelop` die Berechnung immer so früh wie möglich ab – im vornherein für alle Zweige mehr als die minimale HNE berechnen zu lassen scheint mir eine Verschwendung zu sein.

Tabelle 6.9 zeigt die Meßwerte für die gleichen Beispiele wie Tabelle 6.2, wenn man -1 als optionalen Parameter von `develop` wegläßt. Das führt in Charakteristik $\neq 2$ dazu, daß genau eine Transformation T_1 mehr ausgeführt wird als vorher (die HN-Matrix hat im minimalen Fall *einen* unbestimmten Koeffizienten). In Charakteristik 2 dagegen gibt es keinen Unterschied: Der sonst unbestimmte Koeffizient vereinfacht sich zu Null (muß deswegen nicht mit T_1 berechnet werden) und ist daher bereits bei `develop(`p_i`,-1)` bekannt.

Beim Vergleich der beiden Tabellen zeigt sich, daß die Ermittlung des einen unbestimmten Koeffizienten ein Vielfaches der Zeit benötigt, die die Berechnung der ganzen übrigen HNE bis dahin gebraucht hat. Bei p_5 steigt obendrein der Speicherbedarf wesentlich an. p_6 schließlich ist in Charakteristik 0 auf dem Testsystem wegen Speichermangel nicht mehr berechenbar (d. h. der Bedarf liegt bei über 400 Mbyte), in anderer Charakteristik braucht die Berechnung mehrere Tage.

`develop` und `extdevelop` benötigen für ihre Arbeit keine Körpererweiterung. Aus diesem Grund liefern sie ihr Ergebnis in den gleichen Ring zurück, der beim Aufruf aktiv war. Das ist einerseits zwar ein Vorteil (schließlich ist im von `reddevelop` neu erzeugten Ring nichts definiert außer dem Ergebnis; alle anderen Daten befinden sich im alten Ring), andererseits stellt sich hier das Problem, daß das Ergebnis vom temporären Ring, in dem die Berechnung stattfand[10], wieder in den ursprünglichen Ring zurückabgebildet werden muß. Hier kommen wir zum zweiten Punkt, wo der Benutzer durch intelligenten Einsatz von (`ext-`)`develop` selbst Einfluß auf die Performance nehmen kann, denn diese Rückabbildung[11] kann sehr teuer werden.

Eine erste Maßnahme war, den bei Aufruf der Prozedur aktiven Ring zu untersuchen: Ist er schon gleich dem Ring, der für die Berechnung vorgesehen ist (das ist ein Ring mit Variablen `(x,y)` und Ordnung `ls`), kann der Wechsel in einen temporären Ring unterbleiben, und damit entfällt auch die (Hin- und) Rückabbildung. Da alle drei Prozeduren (`develop`, `reddevelop` und `extdevelop`) mit dem gleichen Ring arbeiten, ist diese Voraussetzung bei `extdevelop` automatisch erfüllt, wenn die fortzusetzende HNE mit `reddevelop` (das den Ring permanent wechselt) erzeugt worden ist.

Die zweite Maßnahme betrifft wieder den Singular-Kern: Wenn der aktive Ring ebenfalls zwei Variablen besitzt, läßt sich die Abbildung mit `fetch` statt dem allgemeinen `map` durchführen. Die Effizienz von `fetch` hat sich seit Singular 1.0.0 erheblich verbessert, daher verwenden `develop` und `extdevelop` (wenn der Ring überhaupt gewechselt wurde) vorzugsweise diesen Befehl. Es gibt natürlich immer noch Ringe, bei denen `map` die einzige Möglichkeit ist. Das alles führt dazu, daß man durch die Wahl des Ringes erheblichen Einfluß auf die Zeit für die Abbildung des Ergebnisses nehmen kann; auch die Ordnung des Ringes spielt (bei gleichem Singular-Kommando, besonders bei `map`) eine Rolle.

Nach dem aktuellen Stand ist es nicht mehr so wichtig, den Ringwechsel durch geeignete Wahl des Startringes zu vermeiden. Günstig sind Ringe mit zwei Variablen und Ordnung `ls` sowie alle Ringe mit Ordnung `lp`. Bei diesen spielt die Rückabbildung keine Rolle. Dagegen sollte man auch mit der aktuellen Version von Singular bei aufwendigen Beispielen solche Ringe vermeiden, die mehr als zwei Variablen und Ordnung `ls` haben, sowie alle Ringe mit Ordnung `ds`. Hier braucht die

[10] Der Wechsel in einen Ring mit wohldefinierter Ordnung ist unumgänglich, weil beispielsweise die Berechnung des Newtonpolygons mit dem Singular-Befehl `ord` davon abhängt.

[11] Die Hinabbildung am Anfang ist – außer bei `extdevelop` – unproblematisch, weil das zu entwickelnde Polynom viel kürzer ist als das transformierte innerhalb der zurückzugebenden Entwicklungsliste. Braucht schon die Hinabbildung mehrere Sekunden oder gar Minuten, kann man davon ausgehen, daß sich `develop` und `reddevelop` beim Quadratfreiheitstest totrechnen werden. Bei `extdevelop` haben wir auch ein Auge auf die Abbildung am Anfang zu werfen.

Tabelle 6.9: Testergebnisse von p_4 und p_5 (HN-Matrix ohne undef. Stellen)

	Charakteristik						
	2*	101		32003		0	
$\mathbf{p_4}$							
Speicher (kByte)	197	220		334		408	
Gesamtrechenzeit	0.25 sec	1.05 sec		1.07 sec		1.42 sec	
davon für							
Quadratfreiheit	< 0.02 sec	0.08 sec	8%	0.09 sec	8%	0.17 sec	12%
Irreduzibilitätstests	0.04 sec	0.09 sec	9%	0.09 sec	8%	0.09 sec	6%
Transformation T_1	< 0.02 sec	0.62 sec	59%	0.65 sec	61%	0.89 sec	63%
Transformation T	< 0.02 sec	0.03 sec	3%	0.02 sec	2%	< 0.02 sec	
Newtonpolygon	< 0.02 sec	0.03 sec	3%	0.03 sec	3%	0.06 sec	4%
transform. Polynom							
Anzahl Monome	107	1298		1308		1308	
Anzahl Zeichen	583	10 099		13 357		25 440	
Mehraufwand	0.00 sec	0.61 sec	139%	0.61 sec	133%	0.86 sec	154%
$\mathbf{p_5}$							
Speicher (kByte)	244	1448		2380		3910	
Gesamtrechenzeit	2.4 sec	432.7 sec		504.9 sec		665.6 sec	
davon für							
Quadratfreiheit	< 0.02 sec	0.8 sec	0.2%	0.9 sec	0.2%	0.9 sec	0.1%
Irreduzibilitätstests	0.1 sec	1.6 sec	0.4%	2.1 sec	0.4%	3.0 sec	0.5%
Transformation T_1	2.1 sec	426.7 sec	98.6%	497.8 sec	98.6%	655.5 sec	98.4%
Transformation T	0.04 sec	1.7 sec	0.4%	1.8 sec	0.4%	1.9 sec	0.3%
Newtonpolygon	< 0.02 sec	0.5 sec	0.1%	0.5 sec	0.1%	1.7 sec	0.2%
transform. Polynom							
Anzahl Monome	1887	24 276		30 299		30 299	
Anzahl Zeichen	13 627	221 533		355 847		2 742 643	
Mehraufwand	0.00 sec	345.5 sec	396%	451.1 sec	838%	587.1 sec	748%

* in Charakteristik 2 hat bereits die HN-Matrix der minimalen HNE keine undefinierten Stellen, daher ist diese Spalte nur eine Wiederholung aus Tabelle 6.2.

Tabelle 6.10: Zeiten für die Rückabbildung der HNE von p_5

Zielring	Methode	Zeit und Mehraufwand in			
		Singular 1.0.0		Singular 1.3.0	
`32003,(x,y),ls`	keine Abb.	—	0 %	—	0 %
`32003,(x,y),ds`	`fetch`	174 sec	290 %	20.8 sec	39 %
`32003,(x,y),dp`	`fetch`	18 sec	30 %	1.53 sec	2.8 %
`32003,(x,y),lp`	`fetch`	0.1 sec	0.2 %	0.12 sec	0.2 %
`32003,(x,z),ls`	`fetch`	150 sec	250 %	0.08 sec	0.1 %
`32003,(x,y,z),ls`	`map`	169 sec	282 %	147.7 sec	275 %
`32003,(x,y,z),ds`	`map`	121 sec	202 %	127.3 sec	237 %
`32003,(x,y,z),dp`	`map`	21 sec	35 %	19.2 sec	36 %
`32003,(x,y,z),lp`	`map`	1.5 sec	2.5 %	1.32 sec	2.5 %
`0,(x,y),ls`	keine Abb.	—	0 %	—	0 %
`0,(x,y),ds`	`fetch`	184 sec	207 %	20.4 sec	26 %
`0,(x,y),dp`	`fetch`	23 sec	26 %	1.82 sec	2.3 %
`0,(x,y),lp`	`fetch`	0.3 sec	0.3 %	0.24 sec	0.3 %
`0,(x,z),ls`	`fetch`	232 sec	261 %	0.17 sec	0.2 %
`0,(x,y,z),ls`	`map`	223 sec	251 %	185.1 sec	236 %
`0,(x,y,z),ds`	`map`	191 sec	215 %	152.0 sec	194 %
`0,(x,y,z),dp`	`map`	24 sec	27 %	24.5 sec	31 %
`0,(x,y,z),lp`	`map`	1.3 sec	1.5 %	1.25 sec	1.6 %

Abbildung nämlich teilweise mehr als doppelt so lange wie die ganze eigentliche HNE-Berechnung, so daß sich die gesamte Rechenzeit mehr als verdreifacht.

6.2.6 Faktorisierung

Wenn bei der Faktorisierung des charakteristischen Polynoms ein nichtlinearer Faktor auftaucht, wird bekanntlich in eine Körpererweiterung gewechselt, in der dieser Faktor zerfällt. Anschließend muß das gleiche Polynom über dem neuen Körper faktorisiert werden. Anstatt `factorize` auf das ganze Polynom anzuwenden, erhält man das gleiche Ergebnis auch durch Zerlegung der einzelnen Faktoren, die bereits über dem Teilkörper gefunden wurden. Dabei ist wichtig festzustellen, daß die Nullstellen dieser Faktoren paarweise verschieden sind (wir brauchen die korrekte Vielfachheit e_i und wollen `HN`$(T_2(f), e_i)$ für jede Nullstelle nur einmal aufrufen): Angenommen

$$f \in k[x], \quad f = f_1 f_2 \cdot r \in \tilde{k}[x], \quad \overline{k} \supseteq \tilde{k} \supseteq k, \quad \exists\, a \in \overline{k} :\; x - a \,|\, f_1,\; x - a \,|\, f_2$$

Dann ist wenigstens einer der beiden Faktoren f_1, f_2 über $\tilde{k}[x]$ reduzibel oder beide sind (bis auf Einheiten) gleich, denn $x - a \,|\, \mathrm{ggT}(f_1, f_2) \;\Rightarrow\; \mathrm{ggT}(f_1, f_2) \in \tilde{k}[x]$ ist ein Teiler von f_1, f_2 und keine Einheit.

`factorize` kann also nicht sowohl f_1 als auch f_2 als Faktoren von f zurückgeben.

Es ist natürlich klar, daß diese Maßnahme nur in wenigen Fällen etwas verbessern kann, denn sie greift nur in dem Augenblick, wo es einen Wechsel des Körpers gibt (und mehr als zwei Wechsel im gleichen Beispiel sind sehr selten). Außerdem ist die Faktorisierung nur in Ausnahmefällen kritisch. Ohne die Entdeckung des Beispiels La_4 hätte ich die Änderung gar nicht erst vorgenommen.

In der nachfolgenden Tabelle sind alle Beispiele aus Abschnitt 6.1 aufgeführt, bei denen es mindestens einen Ringwechsel gibt, mit Ausnahme von Ba_5, bei dem (nach dem Ringwechsel) `factorize` nicht eingesetzt wird. Bei den nichtaufgeführten Beispielen kann sich nichts geändert haben.

Zeiten in sec	$Wa_{2,6,8}$	$Wa_{1,\dots,9}$	Ba_6	La_1	La_2	La_3	La_4
`factorize` ohne	0.42	1.17	0.28	0.34	28.5	1.25	13.3 h
und mit Optimierung	0.35	1.10	0.26	0.29	23.6	0.53	3.6 h
Gesamteinsparung	2.1 %	0.007 %	1.2 %	1.2 %	11.5 %	0.2 %	72.6 %

Erwartungsgemäß kommt es nur bei den Beispielen zu einer nennenswerten Verbesserung, wo Polynome mit hohem Grad (13 bei La_2, 18 bei La_4) faktorisiert werden müssen, bei denen im ersten Versuch (vor dem Körperwechsel) irreduzible Bestandteile von ebenfalls hohem Grad übrigbleiben. Dafür lohnt sich die Weiterverwendung der teilweisen Faktorisierung bei La_4 besonders (das Beispiel braucht nur wenig länger als ein Viertel der ursprünglichen Zeit), und zwar weil das charakteristische Polynom gleich dreimal zerlegt wird: erst über $\mathbb{F}_{32003}$, dann über $\mathbb{F}_{32003^7}$ und schließlich über $\mathbb{F}_{32003^{63}}$. Außerdem ist, wie früher schon bemerkt, die Berechnung der HNE nach dieser Faktorisierung fast fertig; träten später noch weitere Polynome auf, die zu faktorisieren sind, würde die Geschwindigkeitssteigerung geringer ausfallen, weil wir deren Faktorisierung über $\mathbb{F}_{32003}$ nicht kennen würden (eine solche existiert im allgemeinen auch gar nicht). Und schließlich ist bei La_4 der Anteil der Faktorisierung an der Gesamtzeit mit 99.4 % (vor der Optimierung 99.8 %) mit Abstand am größten.

6.2.7 Primitives Element

Die Verbesserungen in der Prozedur `primitive` wurden bereits im Teil 5.4 auf Seite 173 vorgestellt: Zuerst testen, ob das zuletzt übergebene Element schon primitiv ist, bei wenigen Körperelementen und zwei Minimalpolynomen eine systematische Suche, sonst Bevorzugung von Linearkombinationen mit kleinen Koeffizienten. Schließlich können wir noch für die Standardbasisberechnung den FGLM-Algorithmus verwenden.

`primitive` kommt natürlich erst zum Einsatz, wenn wir einen Ring erweitern wollen, der bereits ein Minimalpolynom hat. Bei den bisher vorgestellten Beispielen kommt das nur bei Ba_5 und bei La_4 vor, weil dort zweimal in eine Erweiterung gewechselt wird (wir starten immer mit Ringen ohne Minimalpolynom).

Ohne die Verbesserungen dauerten die Ringwechsel bei Ba_5 0.08 Sekunden (das ist 0.02 sec schneller als nachher) und bei La_4 2.9 Sekunden (gegenüber 2.5 sec nach der Verbesserung). Die Verwendung von `fglm` macht sich bei kurzen Beispielen (weniger als 0.2 sec) wegen des zusätzlichen Ringwechsels[12] also noch nicht bezahlt, aber wie beim Quadratfreiheitstest dürfte der zusätzliche Aufwand die Berechnung nur um Sekundenbruchteile verlangsamen können.

Der wahre Nutzen der Änderungen an `primitive` zeigt sich, wenn man das gleiche Beispiel über einer Körpererweiterung startet. Wählt man diese so, daß das charakteristische Polynom die gleichen irreduziblen Faktoren hat wie über $\mathbb{F}_{32003}$, werden von `reddevelop` wieder zwei Ringwechsel ausgeführt, aber diesmal *beide* unter Zuhilfenahme von `primitive`. Außerdem steigt die Komplexität der Berechnung des primitiven Elements durch den höheren Grad des Minimalpolynoms.

Die folgende Tabelle zeigt die Zeiten für den ersten und den zweiten Ringwechsel, wenn man als Startkörper $\mathbb{F}_{32003}[a]/\left\langle a^2+a+2\right\rangle \cong \mathbb{F}_{32003^2}$ bzw. $\mathbb{F}_{32003}[a]/\left\langle a^4-a^3+a+1\right\rangle \cong \mathbb{F}_{32003^4}$ wählt. Weil diese Zeit auch die Abbildung der weiterhin benötigten Daten in den neuen Ring umfaßt, wurde zusätzlich die Zeit gemessen, die allein für die Berechnung des neuen Rings gebraucht wird. Schließlich wurde der Speicher untersucht, der von Singular nach dem Wechsel in den endgültigen Ring belegt war (mit dem Unix-Befehl `top`).

	Rechenzeit		nur für `splitring`		Speicher
	1. Ringw.	2. Ringw.	1. Ringw.	2. Rw.	
$La_4 \in \mathbb{F}_{32003^2}[x,y]$					
vorher	0.10 sec +	115.00 sec	0.08 sec +	63.64 sec	96 839 kB
nachher	0.12 sec +	53.73 sec	0.10 sec +	3.84 sec	1 278 kB
$La_4 \in \mathbb{F}_{32003^4}[x,y]$					
vorher	0.11 sec +	???	0.10 sec +	???	> 395 000 kB
nachher	0.18 sec +	899.22 sec	0.16 sec +	28.74 sec	4 911 kB

Es stellt sich heraus, daß die größte Performancesteigerung im Speicherverbrauch liegt: Weil mit steigendem Grad der Körpererweiterung (im ersten Beispiel $2 \cdot 63 = 126$, im zweiten $4 \cdot 63 = 252$) die Abbildung in den neuen Ring immer mehr Zeit braucht, relativiert sich die Ersparnis für das primitive Element, die beim zweiten Ringwechsel doch beträchtlich war, wieder (der erste Ringwechsel geht wie bei Ba_5 so schnell, daß wir nichts gewinnen). Dagegen wächst der Speicherverbrauch ohne die Verbesserungsmaßnahmen derart stark an, daß das zweite Beispiel wegen des Speichermangels schon nach kurzer Zeit abbricht.

Sowohl Zeit- als auch Speicherersparnis gehen in erster Linie auf den Einsatz des FGLM-Algorithmus zurück – bei $La_4 \in \mathbb{F}_{32003^2}$ bringen die anderen Verbesserungen zusätzlich zu `fglm` sogar eine Verzögerung von einer Zehntelsekunde, weil die Nullstellen des zweiten nichtlinearen Faktors des charakteristischen Polynoms keine primitiven Elemente sind (das heißt natürlich nicht, daß die Berechnung des primitiven Elements nur 0.1 sec gedauert hätte, wenn der Test, ob das zweite Element primitiv ist, gutgegangen wäre. Dieser Test konnte nur darum so schnell sein,

[12] Die Standardbasis wird erst bzgl. Ordnung `dp` berechnet und dann mit `fglm` umgewandelt in eine bzgl. `lp`, dazu muß temporär ein weiterer Ring definiert werden.

weil die Erzeuger des Ideals $\mathfrak{M} := \langle g_1(x), g_2(y) \rangle$ (vgl. Algorithmus `Primitives Element`) bereits eine reduzierte Standardbasis waren. Das ursprünglich gegebene Erzeugendensystem kann aber generell nicht die Eigenschaft von Satz 3.31 (ii) haben, weil einer seiner Erzeuger nur von x abhängt und nicht den Grad 1 hat).

Appendix A

User Manual

A.1 The Library `hnoether.lib`

Prerequisites: This library makes usage of the Singular libraries `primitiv.lib` and `inout.lib`.

Purpose: `hnoether.lib` contains procedures to compute the Hamburger-Noether expansions (HNE) of irreducible plane curve singularities resp. of each branch of a reducible (reduced) plane curve, in any characteristic. These can be used to get further information about the curve:

- invariants of irreducible curves: characteristic exponents and Puiseux pairs (of a complex model), the generators of the semigroup of values, degree of the conductor, delta invariant and the sequence of multiplicities
- invariants of reducible curves: the intersection multiplicity of two branches, the number of quadratic transformations needed to separate two branches, and the sequence of multiplicities
- the initial terms of a parametrization for each branch.

A.1.1 Example Session

Let's suppose an irreducible curve is given by a bivariate polynomial f, say $f \in k[x,y]$. I. e. $R := \overline{k}[[x,y]]/\langle f \rangle$ is the local ring of the curve, and f is an irreducible power series over $\overline{k}$, the algebraic closure of k.

A **Hamburger-Noether expansion** of f is a scheme

$$
\begin{array}{rcl}
z_{-1} & = & a_{0,1}\,z_0 + a_{0,2}\,z_0^2 + \cdots\cdots + a_{0,h_0}\,z_0^{h_0} + z_0^{h_0} z_1 \\
z_0 & = & a_{1,2}\,z_1^2 + a_{1,3}\,z_1^3 + \cdots\cdots + a_{1,h_1}\,z_1^{h_1} + z_1^{h_1} z_2 \\
z_1 & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{i-1} & = & a_{i,2}\,z_i^2 + a_{i,3}\,z_i^3 + \cdots\cdots + a_{i,h_i}\,z_i^{h_i} + z_i^{h_i} z_{i+1} \\
\cdots & = & \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
z_{r-2} & = & a_{r-1,2}\,z_{r-1}^2 + \cdots + a_{r-1,h_{r-1}}\,z_{r-1}^{h_{r-1}} + z_{r-1}^{h_{r-1}} z_r \\
z_{r-1} & = & a_{r,2}\,z_r^2 + a_{r,3}\,z_r^3 + \cdots\cdots\cdots\cdots\cdots\cdots
\end{array}
\qquad (\mathsf{HN})
$$

—where $z_i \in \overline{R}$ are elements of the integral closure of R, $a_{ij} \in \overline{k}$, $r \geq 0$, $h_i > 0$— such that z_{-1} is the residue class of x in R and z_0 the one of y, or vice versa (cf. A. Campillo, "Algebroid Curves in Positive Characteristic", Lect. Notes in Math. 813 (1980), chapt. 2.2).

In Singular (HN) is represented by a list (details later). The first entry is the most important—it's a matrix consisting of the coefficients a_{ij}, the so-called **Hamburger-Noether matrix**.

To compute such a list, we use the following commands:

```
LIB "hnoether.lib";
ring r = 0,(x,y),ls;
poly f = y4-2x3y2-4x5y+x6-x7;
list hne = develop(f);
↦ h(0) = 1
↦ a(1,2) = 1
↦ h(1) = 2
↦ a(2,2) = 1/4
↦ a(2,3) = -1/2
show(hne[1]);        // the Hamburger-Noether matrix
↦ // matrix, 3x3
↦ 0,x, 0,
↦ 0,1, x,
↦ 0,1/4,-1/2
```

Note that the first "non-existing" coefficient a_{i,h_i+1} in each row is marked with an "x", s. t. the values of h_i can also be seen from the matrix. To convert the list into the right hand side of (HN) use

```
displayHNE(hne);
↦ HNE[1]=-y+z(0)*z(1)
↦ HNE[2]=-x+z(1)^2+z(1)^2*z(2)
↦ HNE[3]=1/4*z(2)^2-1/2*z(2)^3
```

Here $-y$ resp. $-x$ is added to the first two lines to indicate that $z_{-1} = z_0 z_1$ corresponds to the residue class of y and $z_0 = z_1^2 + z_1^2 z_2$ to the residue class of x. Now we compute the invariants and an approximate parametrization:

```
displayInvariants(hne);
↦ characteristic exponents  : 4,6,7
↦ generators of semigroup   : 4,6,13
↦ Puiseux pairs             : (3,2)(7,2)
↦ degree of the conductor   : 16
↦ delta invariant           : 8
↦ sequence of multiplicities: 4,2,2,1
param(hne);
↦ // ** Warning:  result is exact up to order 5 in x and 7 in y !
↦ _[1]=1/16x4-3/16x5+1/4x7
↦ _[2]=1/64x6-5/64x7+3/32x8+1/16x9-1/8x10
// if this is not exact enough for us, we can do the following:
list hne_ext = extdevelop(hne,5);
↦ a(2,4) = 47/64
↦ a(2,5) = -61/64
param(hne_ext);
↦ // ** Warning:  result is exact up to order 7 in x and 9 in y !
↦ _[1]=1/16x4-3/16x5+47/128x6-19/32x7+1153/4096x8+379/4096x9-2013/4096x10
   +3721/4096x11
↦ _[2]=1/64x6-5/64x7+237/1024x8-271/512x9+13363/16384x10-14805/16384x11
   +158207/262144x12+8929/32768x13-118401/131072x14+18605/16384x15-226981/262144x16
// or compute a "longer" HNE directly from the beginning:
list hne1 = develop(f,5);
↦ h(0) = 1
↦ a(1,2) = 1
↦ h(1) = 2
↦ a(2,2) = 1/4
↦ a(2,3) = -1/2
↦ a(2,4) = 47/64
↦ a(2,5) = -61/64
param(hne1);
↦ // ** Warning:  result is exact up to order 7 in x and 9 in y !
↦   ......
```

If we wanted to make further use of the invariants, we would say
`list inv = invariants(hne);` instead of `displayInvariants`.
Note that `displayInvariants` and `invariants` will have just the same result for all three HNEs `hne`, `hne_ext` and `hne1`.

Now we come to the case of reducible curves. Here f is allowed to be reducible, but it must be squarefree. If `develop` is given a reducible f, it will in general not produce a HNE, but a message telling you about the reducibility. Thus, if not sure about f, the procedure can still serve to decide whether it is irreducible or not. There is also a procedure `is_irred` to simply answer this question, but it does nothing but silently call `develop`.

```
ring R = 32003,(x,y),dp;
poly f = 100yx24+x24+2y10x12+y20;
list hn = develop(f);
↦ h(0) = 1
↦  The given polynomial is reducible !
is_irred(f);
↦ 0
```

If we want more information about f, we have to use `reddevelop` (this works of course also for irreducible curves):

```
list hne = reddevelop(f);  // or:  list hne = HNdevelop(f);
↦ Change of basering (field extension) necessary!
↦ HNE of one branch found
↦ HNE of one branch found
↦ HNE of one branch found
↦ HNE of one branch found
↦   ~~~ result: 4  branch(es) successfully computed ~~~
size(hne);
↦ 4
```

Here we see that f defines a curve with four branches. `reddevelop` returns a list of four Hamburger-Noether expansions—one for each branch. Note that unlike in the irreducible case, these HNEs need not exist over the originally defined ground field. The message "`Change of basering (field extension) necessary!`" indicates that there was in fact a change to an algebraic extension during the computation. Because of this, `reddevelop` always defines a new ring called "HNEring" over a suitable field—this may also be the original one, if no extension is necessary. There is also a slightly more general procedure called `HNdevelop` that decides which of the two procedures (`develop` or `reddevelop`) to use. In this example (as nearly always) it would have exactly the same result as `reddevelop`.

```
basering;
↦ //   characteristic : 32003
↦ //   1 parameter    : a
↦ //   minpoly        : (a2+1)
↦ //   number of vars : 2
↦ //        block   1 : ordering ls
↦ //                  : names    x y
↦ //        block   2 : ordering C
```

`hne[1]`,…,`hne[4]` can be treated like the output of `develop`, e. g.
`displayInvariants(hne[1]);` shows the curve invariants of the first branch,
`displayHNE(hne[4]);` shows the HNE of the last branch,
`param(extdevelop(hne[4],20));` computes a quite good parametrization etc.
Additionally do we have the following opportunities:

`displayInvariants` and `displayHNE` are also applicable to the whole list of HNEs—in this case the data is shown branchwise. `displayInvariants` computes moreover a table of all intersection multiplicities:

```
displayInvariants(hne);
↦ --- invariants of branch number 1 : ---
↦ characteristic exponents  : 5,6
↦ generators of semigroup   : 5,6
↦ Puiseux pairs             : (6,5)
↦ degree of the conductor   : 20
↦ delta invariant           : 10
↦ sequence of multiplicities: 5,1
↦
↦ --- invariants of branch number 2 : ---
↦ characteristic exponents  : 5,6
       ⋮
↦ ------------- intersection multiplicities : -------------
↦
↦ branch |    4     3     2
↦ -------+-----------------
↦     1  |   30,   30,   33
↦     2  |   30,   30
↦     3  |   33
```

Then we have procedures computing invariants of reducible curves:

```
intersection(hne[1],hne[2]); // intersect. multiplicity of 1st & 2nd branch
↦ 33
displayMultsequence(hne);     // sequence of multiplicities (reducible case)
↦ [(5,5,5,5)],
↦ [(1,1,1,1)],
↦ [(1,1,1,1)],
↦ [(1,1,1,1)],
↦ [(1,1,1,1)],
↦ [(1,1,1,1)],
↦ [(1,1),(1,1)],
↦ [(1,1),(1,1)],
↦ [(1,1),(1,1)],
↦ [(1),(1),(1),(1)]
// number of quadratic transformations needed to separate branches 1 & 2:
separateHNE(hne[1],hne[2]);
↦ 9
```

Again (as with `displayInvariants`), for the further processing of the sequence of multiplicities, there also exists a procedure `multsequence`, which has this sequence (in another representation) as return value.

A.1.2 Hints to Handle Large Computations Efficiently

The thing one usually is most interested in is to compute the curve invariants. If you don't need the parametrization, you sometimes can save a lot of time (and memory) by taking the following suggestions into consideration:

- Use `develop(f,-1)`—this will usually leave undefined entries in the Hamburger-Noether matrix, but stop the computation as soon as we have enough data to determine the invariants.
- If you want to keep a HNE, but don't intend to extend it with `extdevelop`, replace the output `hne` of `develop` or `reddevelop` by `stripHNE(hne)` (only necessary if `hne` consumes much memory).
- No matter whether you're interested in the parametrization or not, prefer a ring of the form `ring ...,(x,y),ls` if you use `develop` or `extdevelop`. There's no need to worry about `reddevelop`—it defines a ring which has exactly these properties (hence `extdevelop` will be efficient with `HNEring` as basering). The choice of the ordering and the number of variables is more important than their names; avoid the ordering `ds`, and in rings with more than two variables prefer `lp`.

Example (this was done on a HP-UX 9000/735/125 MHz machine, times in seconds):

```
ring r = 0,(a,b),ds;
poly p_1 = b2+a3;
poly p_2 = p_1^2 + a5b;
poly p_3 = p_2^2 + a^10 *p_1;
poly p_4 = p_3^2 + a^20 *p_2;
poly p_5 = p_4^2 + a^40 *p_3;
timer;
↦ 0
list hne = develop(p_5);
↦ h(0) = 1
↦ a(1,2) = -1
↦ h(1) = 2
↦ a(2,2) = -1
↦ h(2) = 2
↦ a(3,2) = -1
↦ h(3) = 2
↦ a(4,2) = -1
↦ h(4) = 2
↦ a(5,2) = -1
↦ a(5,3) = 10
timer;
↦ 678
kmemory();
```

```
↦ // memory used by active variables (kilobyte):
↦ 2825
displayInvariants(hne);
↦ characteristic exponents  : 32,48,56,60,62,63
↦ generators of semigroup   : 32,48,104,212,426,853
↦ Puiseux pairs             : (3,2)(7,2)(15,2)(31,2)(63,2)
↦ degree of the conductor   : 1612
↦ delta invariant           : 806
↦ sequence of multiplicities: 32,16,16,8,8,4,4,2,2,1
timer;
↦ 679
kill hne;
list hne = develop(p_5,-1);
↦ h(0) = 1
↦ a(1,2) = -1
↦ h(1) = 2
     ⋮
↦ h(4) = 2
↦ a(5,2) = -1
"time:",timer-679; kmemory();
↦ time:  114
↦ // memory used by active variables (kilobyte):
↦ 1725
ring s = 0,(x,y),ls;
poly p_5 = fetch(r,p_5);
kill r;
timer;
↦ 793
list hne = develop(p_5,-1);
↦ .........
"time:",timer-793; kmemory();
↦ time:  87
↦ // memory used by active variables (kilobyte):
↦ 1725
hne = stripHNE(hne);
"time:",timer-793-87; kmemory();
↦ time:  0
↦ // memory used by active variables (kilobyte):
↦ 308
displayInvariants(hne);    // of course the same as above
↦ characteristic exponents  : 32,48,56,60,62,63
↦ generators of semigroup   : 32,48,104,212,426,853
     ⋮
timer-793-87;
↦ 0
```

This is of course already an extreme example, but it shows that there are cases where the choice of the ring and a restriction to the minimal data needed can really make a big difference. By the way, the time you save by taking a "stripped" HNE as parameter for `invariants`, `param` etc. normally compares with the time it takes to invoke `stripHNE`; so concerning the computation time this procedure becomes only important if you intend to call more than one proc with the same HNE.

A.1.3 Documentation of the Important Procedures

```
LIBRARY:  hnoether.lib   PROCEDURES FOR THE HAMBURGER-NOETHER DEVELOPMENT

          Important procedures:
HNdevelop(f);       Hamburger-Noether development of f
develop(f [,n]);    Hamburger-Noether development of irreducible curves
reddevelop(f);      Hamburger-Noether development of reducible curves
extdevelop(hne,n);  extension of the Hamburger-Noether development hne of f
param(hne);         returns a parametrization of f (input=output(develop))
displayHNE(hne);    display Hamburger-Noether development as an ideal
invariants(hne);    invariants of f, e.g. the characteristic exponents
displayInvariants(hne);  display invariants of f
intersection(hne1,hne2); intersection multiplicity of two curves
multsequence(hne);  sequence of multiplicities
displayMultsequence(hne);  display sequence of multiplicities
stripHNE(hne);      reduce amount of memory consumed by hne
is_irred(f);        test for irreducibility

          perhaps useful procedures:
puiseux2generators(m,n); convert Puiseux pairs to generators of semigroup
multiplicities(hne);     multiplicities of blowed up curves
separateHNE(hne1,hne2);  number of quadratic transf. needed for separation
squarefree(f);      returns a squarefree divisor of the poly f
allsquarefree(f,l); returns the maximal squarefree divisor of the poly f
newtonpoly(f);      Newton polygon of polynom f
getnm(f);           intersection points of the Newton polygon with the axes
T_Transform(f,Q,N); returns f(y,xy^Q)/y^NQ (type f(x,y): poly, Q,N: int)
T1_Transform(f,d,M); returns f(x,y+d*x^M)  (type f(x,y): poly, d:number, M:int)
T2_Transform(f,d,M,N,ref);   a composition of T1 & T
koeff(f,I,J);       gets coefficient of indicated monomial of poly f (I,J:int)
redleit(f,S,E);     restriction of monomials of f to line (S-E)
leit(f,n,m);        special case of redleit (for irred. polynomials)
```

For examples to the procedures look at the preceding example sessions or use the online help (type "`LIB "hnoether.lib"; example` <procedure>;" in Singular).

Procedures Computing a Hamburger-Noether Expansion

HNE for irreducible curves: `develop`

USAGE: `develop(f [,n]);` `f` polynomial in two variables, `n` integer

ASSUME: `f` irreducible as a power series (for reducible `f` use `reddevelop`)

RETURN: Hamburger-Noether development of `f`, in form of a `list`:

`[1]`: Hamburger-Noether `matrix`:
each row contains the coefficients of the corresponding line of the Hamburger-Noether expansion (HNE). The end of the line is marked in the matrix as the first ring variable (usually x).

`[2]`: `intvec` indicating the length of lines of the HNE

`[3]`: `int`:
- 0 if the 1st ring variable was transversal (with respect to `f`),
- 1 if the variables were changed at the beginning of the computation
- -1 if an error has occurred

`[4]`: the transformed polynomial of `f` to make it possible to extend the Hamburger-Noether development a posteriori without having to do all the previous calculation once again (0 if not needed).

`[5]`: `int`:
- 1 if the curve has exactly one branch (i. e. is irreducible)
- 0 else (i. e. the curve has more than one HNE or `f` is not valid)

DISPLAY: the (non zero) elements of the HNE (if not called by another procedure)

NOTE: If the optional parameter `n` is given, the HN matrix will have at least `n` columns. Otherwise the number of columns will be chosen minimal s. t. the matrix contains all necessary information (i. e. all lines of the HNE but the last (which is in general infinite) have place). If `n` is negative, the algorithm is stopped as soon as possible, i. e. the information computed is enough for `invariants`, but the HN matrix may contain undetermined elements, which are marked with the 2nd variable (of the basering).

In any case, the optional parameter only affects the calculation of the *last* line of the HNE; `develop(f)` gives already all necessary information for the procedure `invariants`. A negative value of n will result in a very poor parametrization, but it can make `develop` faster; a positive value will improve the exactness of the parametrization.

For time critical computations it is recommended to use "`ring ...,(x,y),ls`" as basering—it increases the algorithm's speed.

Also note that the assumption "`f` irreducible" is not a priori necessary—this will be checked during the computation. If `f` is not squarefree, a warning appears and `develop` continues with a squarefree divisor of `f` (normally the radical). If `f` (resp. the squarefree divisor `develop` has taken) is reducible, the algorithm will break as soon as this comes out. In this case the 5th entry of the result (cf. RETURN) is 0. If by chance `develop` is able to return a HNE, it will do this (but there are of course more than only one), otherwise the 3rd entry will be -1.

`f` can be a member of a great variety of rings. The admissible ground fields are shown in table A.1. Names of variables and monomial orderings don't matter (but may influence the computation time—cf. section A.1.2). If the number of variables is greater than two, `f` must depend only on the first and second—otherwise all monomials containing other variables are replaced by zero. There is one difference with additional variables: If the basering has more than two variables, `param` will later choose the third one instead of the first to represent the parametrization of `f` (this variable perhaps being less confusing).

It is by the way possible to make `develop` as well as the other HNE computing procedures perform their task more quietly: Set the Singular variable `printlevel` to a negative value, and all messages like "The polynomial is reducible!" should disappear.

HNE for reducible curves: `reddevelop`

USAGE: `reddevelop(f);` `f` polynomial in two variables

RETURN: Hamburger-Noether development of `f`:
A `list` of lists in the form of `develop(f)` (`matrix,intvec,int,poly`— the last `int` indicating irreducibility is omitted); each entry contains the data for one of the branches of `f`. For more details see the description of `develop`.

CREATE: a ring with name "HNEring", variables x, y and ordering `ls` over a field extension of the current basering's ground field. As the Hamburger-Noether development of a reducible curve normally does not exist in the given basering, `reddevelop` always defines HNEring and changes to it. The field extension is chosen minimal.

Again there will appear a warning if `f` is not squarefree. But this time, if the global variable `printlevel` > 0, you have the choice whether the continuation with a squarefree divisor is sufficient for you or not. In the former case it may be that factors of the form g^p (p being the characteristic of the ring) are lost, to the effect that the corresponding branches do not appear in the result of the procedure. In the latter case you can force `reddevelop` to continue with the radical of `f`, what ensures that you get a HNE for really *every* branch of the curve. The price for this is that a bivariate factorization is used to compute the radical, so it can be that you'll have to wait quite a long time. It's also possible to quit the procedure and recall it with a better argument (e. g. if you have a better possibility to get the radical of `f` yourself). As procedures should by default not be interactive, it aborts if `printlevel` ≤ 0 and `f` is not squarefree.

Observe that the high-cost computation of the radical would not make any sense with `develop`, as long as we get at least *one* squarefree factor.

Concerning the rings containing `f` most of the explanations for `develop` apply also here. There are only some differences with the admissible ground fields (cf. table A.1), and as `reddevelop` always changes to "HNEring", the names of the variables and the monomial ordering are now completely irrelevant (also for the computation's speed).

General procedure for HNEs: `HNdevelop`

USAGE: `HNdevelop(f);` `f` polynomial in two variables

RETURN: Hamburger-Noether development of `f`: The same as `reddevelop(f)`, whenever `HNdevelop` could be replaced by `reddevelop` (see below). This means it gives a `list` of lists in the form of `develop`.

Table A.1: **admissible ground fields**
for computing the Hamburger-Noether expansion

	annotations	0	(0,a)	(0,t [,..])	real	p	(p,a)	(p^k,g)	(p,t [,..])
`develop`	only irred. curves no ring change	✓	✓	✓	–	✓	–	✓	–
`reddevelop`	red. & irred. curves ring change	✓*	–*	(✓)	–	✓	✓	–	(✓)
`extdevelop`	one branch no ring change	✓	✓	✓	–	✓	✓	✓	✓

legend: ✓: ok – : not possible

(✓): Singular doesn't allow algebraic extensions of this ground field. As long as such an extension is not needed, everything works fine. Otherwise the HNEs not existing over the given field will be dropped. Note that there still can be a non empty result (the user is notified if some HNEs are missing).

* : As long as `factorize` is not implemented in Singular for algebraic extensions over $\mathbb{Q}$, `reddevelop` only works for $\mathbb{Q}$ as long as no algebraic extension is needed and doesn't work for $\mathbb{Q}(a)$. This may change in future versions of Singular—after that both fields will be ok.

p : prime number, k : integer greater than one
a : algebraic element (i. e. this ring has a minimal polynomial)
t [,..] : one or more transcendent elements (i. e. ring is without minimal polynomial)
g : a generator of the group of unities of GF(p^k)

CREATE: a ring with name "HNEring". It is the same ring as the one created by `reddevelop`.

NOTE: `HNdevelop` decides which procedure (`develop` or `reddevelop`) applies best to the given problem and calls it. Whenever possible, this is of course the latter one, because we cannot know in advance whether `f` is irreducible or not, and hence the procedure always performs a ring change. The following table shows which procedure is used in which case.

ground field		procedure called by `HNdevelop`	
0		`reddevelop`	
(0,a)	a algebraic	`develop`	(this may change in the future)
(0,t [,..])	t transcendent	`reddevelop`	
real		——	(not admissible)
p	prime number	`reddevelop`	
(p,a)	a algebraic	`reddevelop`	
(p^k,g)	$k > 1$	`reddevelop`	after a ring change to (p,a)
(p,t [,..])	t transcendent	`reddevelop`	

If not all existing HNEs could be found (in algebraic extensions of $\mathbb{Q}$ this will most often lead to an empty list as return value), the user will be notified.

Extension of a HNE: `extdevelop`

USAGE: `extdevelop(l,n);` `list l` (`matrix m`, `intvec`, `int`, `poly` [, `int`]), `int n` where `l` is the output of `develop(f)` (or `extdevelop(L,N)`, or one entry of the output of `reddevelop(f)`)

RETURN: an extension of the Hamburger-Noether development of `f` in the same form as `l`, i. e. a `list L` (`matrix m'`, `intvec`, `int`, `poly`)

The new HN matrix `m'` will have at least `n` columns (if the HNE isn't finite). Thus if `f` is irreducible, `extdevelop(develop(f),n);` (in most cases) will produce the same result as `develop(f,n)`.
See the description of `develop` for more details.

NOTE: If the matrix `m` of `l` has N columns, the exactness of `param(extdevelop(l,n))` will be increased by at least ($\mathtt{n} - N$) more significant monomials compared with `param(l)`.

`extdevelop` can extend a HNE (but not a list of HNEs!) in an arbitrary ring (as long as it doesn't contain floating point numbers, cf. table A.1), so if `l` was computed successfully by one of the three procedures in this section, the ring in which `l` is returned is automatically correct. Remember that the computation time may depend on the variable names and the ordering. So if it took already long to compute `l` with `develop` or `extdevelop` it could make sense to define a new ring and copy[1] `l` over to it.

Procedures Making Use of Hamburger-Noether Expansion(s)

Show a HNE on screen: `displayHNE`

There are two reasonable ways to inspect a Hamburger-Noether expansion `hne`: With `print(hne[1]);` or `show(hne[1]);` you can look at the Hamburger-Noether matrix and see how the HNE looks like. This has a very compact form and is therefore most useful for complex HNEs. Or you use `displayHNE` to compute the right hand side of the scheme (HN) on page 204:

USAGE: `displayHNE(l [,n] );` `l` is a `list` (either a HNE, output of (`ext`)-`develop`, or a list of HNEs, output of `reddevelop`) and `n` is `int`

RETURN:

▷ If only one argument is given, no return value, but display an ideal `HNE` of the following form:

```
HNE[1]=-y +□*z(0)^1 +□*z(0)^2 +...+ z(0)^◊*z(1)
HNE[2]=-x +          □*z(1)^2 +...+ z(1)^◊*z(2)
HNE[3]=              □*z(2)^2 +...+ z(2)^◊*z(3)
.......              ........................
HNE[r+1]=            □*z(r)^2 +□*z(r)^3 +.....
```

[1] make sure you map the 1st variable of the old ring onto the 1st of the new one etc.

where `x`,`y` are the indeterminates of the basering. The values of □ are the coefficients of the Hamburger-Noether matrix, the values of ◊ are represented in the HN matrix as "`x`".
The 1st line (`HNE[1]`) means that $y = \square * z(0)^1 + \dots$, the 2nd line (`HNE[2]`) means that $x = \square * z(1)^2 + \dots$, so you can see which indeterminate corresponds to which line (it's also possible that x corresponds to the 1st line and y to the 2nd).

▷ If a second argument is given, create and export a new ring with name "`displayring`" containing an ideal "`HNE`" as described above.

▷ If `l` contains a list of HNEs, `displayHNE` shows each of them in the form described above. The optional parameter is then ignored.

Parametrization of a curve or a branch: `param`

USAGE: `param(hne);` `list hne` representing a HNE
`param` takes the output `hne` of `develop(f);` and gives a parametrization for `f`; the first variable of the active ring is chosen as indefinite. If the ring contains more than two variables, the 3rd variable is chosen (remember that `develop` takes the first two variables and therefore the other variables should be unused).

RETURN: `ideal` of two polynomials: if the HNE was finite, `f(param[1],param[2])` will be zero. If not, the real parametrization will be two power series; `param` will then return two polynomials with the same initial terms.

Invariants of irreducible curves: `invariants`

USAGE: `invariants(l);` `list l` representing a HNE
`invariants` takes the output `hne` of `develop(f);` and computes the following irreducible curve invariants of `f`:

RETURN: a `list`, if `l` contains a valid HNE:

`invariants(l)[1]`:	`intvec`	characteristic exponents
`invariants(l)[2]`:	`intvec`	generators of the semigroup
`invariants(l)[3],[4]`:	2× `intvec`	Puiseux pairs, 1st and 2nd components
`invariants(l)[5]`:	`int`	degree of the conductor
`invariants(l)[6]`:	`intvec`	sequence of multiplicities

If an error occurred in the procedure `develop`, `invariants` returns the empty list.

Display invariants of irreducible curves: `displayInvariants`

USAGE: `displayInvariants(l);` `list l` representing a HNE or a list of HNEs
`l` can be the output both of `develop(f);` and of `reddevelop(f);`

DISPLAY: invariants of the corresponding branch, resp. of all branches, in a better readable form.

If `l` is the output of (`ext`)`develop`, the procedure shows those invariants which are computed by `invariants`. If it is the output of `reddevelop`, `displayInvariants` does this for all branches of `f` and shows in addition the intersection multiplicity of each pair of branches.

RETURN: nothing

Intersection multiplicity of two curves: `intersection`

USAGE: `intersection(hne1,hne2);` `hne1`, `hne2` two `lists` representing a HNE (normally two entries out of the output of `reddevelop`)

RETURN: intersection multiplicity (of type `int`) of the branches/curves corresponding to `hne1` and `hne2`

Sequence of multiplicities: `multsequence`

USAGE: `multsequence(l);` `list l` representing a HNE or a list of HNEs
`l` can be the output both of `develop(f);` and of `reddevelop(f);`

RETURN:

- ▷ If `l` = `develop(f)` or `l` = `reddevelop(f)[i]` :
  `intvec` of the sequence of multiplicities of the curve resp. of the `i`-th branch (the same as `invariants(l)[6];`)
- ▷ If `l` = `reddevelop(f)` : `list` of two integer matrices:

`multsequence(l)[1][i,*]` contains the multiplicities of the branches at their infinitely near point of 0 in its (`i` − 1) order neighbourhood (i. e. `i` = 1: multiplicities of the branches themselves,
`i` = 2: multiplicities of their 1st quadratic transformed etc.,
or to say it in another way: `multsequence(l)[1][*,j]` is the sequence of multiplicities of the `j`-th branch).

`multsequence(l)[2][i,*]` contains the information which of these infinitely near points coincide.

NOTE: The order of elements of the list obtained from `reddevelop` *must not* be changed (because then the coincident infinitely near points probably couldn't be grouped together, cf. the example for `displayMultsequence` on page 207). Hence it is not wise to compute the HNE of several polynomials separately, put them into a list `l` and call `multsequence(l)`.

Use `displayMultsequence` to produce a better readable output for reducible curves on the screen.

Display sequence of multiplicities: `displayMultsequence`

USAGE: `displayMultsequence(l);` `list l` representing a HNE or a list of HNEs, normally the output of `develop(f);` or `reddevelop(f);`

DISPLAY: the sequence of multiplicities:

- ▷ If `l` = `develop(f)` or `l` = `reddevelop(f)[i]` :
  just the sequence $a, b, c, \ldots, 1$
- ▷ If `l` = `reddevelop(f)` : a sequence of the following form:

$$\begin{array}{ll} \left[\,(a_1, \ldots, b_1, \ldots, c_1)\,\right], & \\ \left[\,(a_2, \ldots), \ldots, (\ldots, c_2)\,\right], & \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots & \\ \left[\,(a_n), (b_n), \ldots, (c_n)\,\right] & \qquad \text{with} \end{array}$$

$a_1, \ldots, a_n$: the sequence of multiplicities of the 1st branch,
[...]: the multiplicities of the j-th transformed of all branches,
(...): the multiplicities of those branches meeting at the same infinitely near point, and $a_n = b_n = \ldots = c_n = 1$.

RETURN: nothing

NOTE: The same restrictions for `l` as in `multsequence` apply also here.

Reduce the information in HNEs: `stripHNE`

USAGE: `stripHNE(l);` `list l` representing a HNE or a list of HNEs
`l` can be the output both of `develop(f);` and of `reddevelop(f);`

RETURN: a `list` in the same format as `l`, but the 4th entry (of `l` resp. each component of `l`) is replaced by zero (cf. RETURN value of `develop`).

NOTE: The purpose of this procedure is to remove (possibly huge amounts of) data no longer needed. It is useful if one or more of the polynomials in `l` consumes much memory. It is still possible to compute invariants, parametrizations etc. with the "stripped" HNE(s), but it is not possible to use `extdevelop` with them (read also subsection A.1.2).

Multiplicities during blowing up: `multiplicities`

USAGE: `multiplicities(l);` `list l` representing a HNE
`l` normally is the output of `develop(f);`

RETURN: `intvec` of the different multiplicities that occur during the successive blowing up of the curve corresponding to `f`.

NOTE that `multiplicities(l)` is in general not the *sequence* of multiplicities! (That's why this procedure is not so important).

Separate two branches of a curve: `separateHNE`

USAGE: `separateHNE(hne1,hne2);` `list hne1,hne2` representing a HNE
`hne1`,`hne2` normally are two components of `reddevelop(f);`

RETURN: `int`: The number of quadratic transformations needed to separate both branches of `f` (more general: both irreducible curves) given by their HNEs `hne1` and `hne2`.

A value of 1 indicates that both branches intersect transversally, otherwise they're tangent.

Other Procedures

The following procedures are primarily designed for usage by the procedures explained above, but may also be useful on their own. Note that for performance reasons some of them make special assumptions and are not as general as they could be. The order reflects the author's opinion about their being of service.

Compute semigroup out of Puiseux pairs: `puiseux2generators`

USAGE: `puiseux2generators (m,n);` `m,n` `intvecs` with 1st resp. 2nd components of Puiseux pairs (same format as `invariants(l)[3]` & `[4]`)

RETURN: `intvec` of the generators of the semigroup of values

Is a power series irreducible? `is_irred`

USAGE: `is_irred(f);` `f` a squarefree bivariate polynomial

RETURN: an integer:

`is_irred(f)` $= 1$ if `f` is irreducible as a formal power series over the algebraic closure of its coefficient field (`f` defines an analytically irreducible curve at zero)

`is_irred(f)` $= 0$ else

NOTE: `f` $= 0$ and units in the ring of formal power series are not considered irreducible. `is_irred` makes an internal call of `develop`. So the same prerequisites for the ground field apply also here (see table A.1). The information about the irreducibility is part of the result of `develop`, too—`is_irred` just gives you the 5th entry, s. t. you can equally call `develop` yourself to answer the question. Another possibility would be to use `reddevelop` or `HNdevelop` and to count the number of HNEs that are computed, but if `f` is reducible, this will cost more time and memory.

Make a power series squarefree: `squarefree`

USAGE: `squarefree(f);` `f` a bivariate `polynomial`

RETURN: a squarefree divisor of `f`

Normally the return value is a greatest squarefree divisor, but there is an exception: factors with a p-th root, p the ring characteristic, are lost.

For most coefficient fields k of `f` we have a nice property:

$$k \in \{\mathbb{Q},\, \mathbb{F}_p,\, \mathbb{F}_p(a) \mid p \text{ prime number},\ a \text{ algebraic}\} \quad \Longrightarrow$$
$$\big(\mathtt{f} \text{ squarefree as polynomial} \iff \mathtt{squarefree(f)} = \mathtt{f}\big)$$

so `squarefree` allows to decide whether `f` is squarefree or not.

If $k = \mathbb{Q}(a)$, a algebraic, we still have

$$\mathtt{f} \text{ squarefree as power series} \iff \frac{\mathtt{f}}{\mathtt{squarefree(f)}} \text{ is a unit in } k[\![x,y]\!]$$

If $k = \mathbb{F}_{p^l}$ (i.e. k was defined by its number of elements instead of giving a minimal polynomial), more factors may be lost. It's only guaranteed that the result is squarefree, divides `f` and—if there's no $g \in k[x,y]$ s.t. $\mathtt{f} = g^p$ —has a degree strictly bigger than zero.

Obtain all squarefree factors: `allsquarefree`

USAGE: `allsquarefree(f,l);  poly f,l`
`l` is the squarefree divisor of `f` you get by `l=squarefree(f);`

RETURN: the squarefree divisor of `f` consisting of *all* irreducible factors of `f`

ASSUME: The Singular commands `gcd` and `factorize` must be implemented for `f` (i.e. for bivariate polynomials in the ring being active).

NOTE: This procedure uses `factorize` to get the missing factors of `f` not contained in `l` and therefore may be *slow*.

Newton polygon: `newtonpoly`

USAGE: `newtonpoly(f);` `f` bivariate polynomial

RETURN: `list` of `intvec`(x,y) of coordinates of the Newton polygon of `f`

NOTE: `newtonpoly` is designed for a special case (to improve speed):

– The basering must have ordering `ls`
– $\mathtt{f}(x,0) \neq 0 \neq \mathtt{f}(0,y), \quad \mathtt{f}(0,0) = 0$

i.e. the Newton polygon has two different endpoints which both lie on the coordinate axes.

Intersect Newton polygon with axes: `getnm`

USAGE: `getnm(f);` `f` bivariate polynomial

RETURN: `intvec`(n,m) : $(0,n)$ is the intersection point of the Newton polygon of `f` with the `y`-axis, $n = -1$ if it doesn't exist,
$(m,0)$ is its intersection point with the `x`-axis, $m = -1$ if this point doesn't exist.

ASSUME: ring has ordering `ls` or `ds`.

The following three procedures define the main polynomial transformations needed to compute the Hamburger-Noether expansion:

Transformation T: `T_Transform`

USAGE: `T_Transform(f,Q,N);  poly f,  int Q,N`

RETURN: $$T(\mathtt{f})(x,y) = \frac{\mathtt{f}\left(y, xy^{\mathtt{Q}}\right)}{y^{\mathtt{NQ}}}$$ (only useful if `f` is irreducible)

Transformation T_1: `T1_Transform`

USAGE: `T1_Transform(f,`δ`,Q);` `poly f,` `number` δ, `int Q`

RETURN: $T_1(\mathtt{f})(x,y) = \mathtt{f}\left(x, y+\delta x^{\mathtt{Q}}\right)$

Transformation T_2: `T2_Transform`

USAGE: `T2_Transform(f,`δ`,M,N,ref);` `poly f,ref,` `number` δ, `int M,N`

ASSUME: `ref` has the same Newton polygon as f (but can be simpler).

RETURN: a `list` of two elements: The first is `poly` $T_2(\mathtt{f},\delta',\mathtt{M},\mathtt{N})$, the second one is `number` $\delta' \in \{\delta,\, 1/\delta\}$

The exact definition of T_2 is definition 3.12 (page 77). It is a composition of a certain number of transformations T (modified for the reducible case) and one transformation T_1 and looks like

$$T_2(f,\delta',M,N) = T_1\left(\frac{f\left(x^\alpha y^\tau, x^\beta y^\sigma\right)}{x^s y^t},\, \delta',\, Q\right)$$

where $\alpha,\beta,\sigma,\tau,s,t \in \mathbb{N}$, s and t maximal s.t. the quotient $f(\ldots)/x^s y^t$ is still a polynomial. Q is the last quotient in Euclid's algorithm applied to M and N, and the value of δ' depends on the number of quotients occurring in that algorithm.

For `ref` you can e. g. simply use `ref` := `f` again or (better)
`ref` := `referencepoly(newtonpoly(f))`.

Pick coefficient out of a polynomial: `koeff`

USAGE: `koeff(f,I,J);` `poly f` bivariate, `int I,J`

RETURN: If $\mathtt{f} = \sum_{i,j} a_{i,j}\, x^i y^j$, then `koeff(f,I,J)` $= a_{\mathtt{I},\mathtt{J}}$ (of type `number`)

NOTE: `J` must be in the range of the exponents of y

Quasihomogeneous leading form: `redleit`

USAGE: `redleit(f,S,E);` `f poly,` `S,E intvec`(x,y): two different points on a line in the Newton diagram of `f`

RETURN: a polynomial consisting of all monomials of `f` which lie on or below that line

NOTE: If the line defined by `S` and `E` is part of the Newton polygon, the result is the quasihomogeneous leading form of `f` wrt. that line.

Quasihomogeneous leading form (irreducible case): `leit`

USAGE: `leit(f,n,m);` `poly f,` `int n,m`

RETURN: all monomials on the line from $(0,\mathtt{n})$ to $(\mathtt{m},0)$ in the Newton diagram

A.2 The Library `primitiv.lib`

Prerequisites: This library makes usage of the Singular library `random.lib`.

Purpose: `primitiv.lib` contains procedures to compute a primitive element of algebraic extensions and to define algebraic extensions of rings.

A.2.1 Example Session

We want to compute a number α s. t. $\mathbb{Q}\left(\sqrt{2}, \sqrt{3}, i\right) = \mathbb{Q}(\alpha)$

```
LIB "primitiv.lib";
ring r = 0,(x,y,z),dp;
ideal i = x2-2, y2-3, z2+1;
ideal j = primitive(i);
j;[2]
↦ j[1]=z8-16z6+88z4+192z2+144
↦ j[2]=1/72z7-35/144z5+14/9z3+13/12z
↦ j[3]=-7/384z7+59/192z5-181/96z3-21/16z
↦ j[4]=5/1152z7-37/576z5+95/288z3+59/48z
// The first entry is the minimal polynomial of the primitive element α,
// the rest tells us how to get the former generators. We can verify this:
ring s = (0,a),x,dp;
minpoly = a8-16a6+88a4+192a2+144;
map T = r,0,0,a;
ideal j = T(j);
j;
↦ j[1]=0
↦ j[2]=(1/72a7-35/144a5+14/9a3+13/12a)
↦ j[3]=(-7/384a7+59/192a5-181/96a3-21/16a)
↦ j[4]=(5/1152a7-37/576a5+95/288a3+59/48a)
j[2]^2-2;
↦ 0
j[3]^2-3;
↦ 0
j[4]^2+1;
↦ 0
```

Now we consider two algebraic elements $\alpha, \beta \in \overline{\mathbb{F}_7}$ given by their minimal polynomials $f_\alpha = x^3 + x + 1$, $f_\beta = x^6 + 3x^4 - 3x^2 + 1 \in \mathbb{F}_7[x]$. The first step is to find a minimal polynomial $f'_\beta \in \mathbb{F}_7(\alpha)[x]$ for β $\big($in the above example this could be omitted because we knew that $x^2 - 3$ has no root in $\mathbb{Q}(\sqrt{2})$, nor does $x^2 + 1$ have one in $\mathbb{Q}(\sqrt{2}, \sqrt{3})\big)$.

[2] The result contains random data (cf. p. 224)—don't be surprised by different coefficients.

```
ring F7a = (7,a),x,dp;
minpoly = a3+a+1;
factorize(x6+3x4-3x2+1);
↦ [1]:
↦    _[1]=1
↦    _[2]=x2+(2a2)
↦    _[3]=x2+(-2a2-a+2)
↦    _[4]=x2+(a+1)
↦ [2]:
↦    1,1,1,1
```

// we assume that β satisfies $\beta^2 = -2\alpha^2$, i.e. $f'_\beta = x^2 + 2\alpha^2$:

```
ring F7 = 7,(x,y),dp;
ideal i = x3+x+1, y2+2x2;
ideal j = primitive_extra(i);
j;
↦ j[1]=y6+3y4-3y2+1
↦ j[2]=-2y4-3y2
primitive(i);
↦ _[1]=y6+3y4-3y2+1
↦ _[2]=-2y4-3y2
↦ _[3]=y
ring F7b = (7,b),x,dp;
minpoly = b6+3b4-3b2+1;
number a = -2b4-3b2;
a^3+a+1;
↦ 0
b2+2*a^2;
↦ 0
```

Note that in this example β was already a primitive element. Both procedures offer the possibility to reconstruct f_β from f'_β and to express α in terms of β.

Finally do we come to algebraic ring extensions:

```
ring r1;
factorize(x2+1,1);
↦ _[1]=x2+1
splitring(x2+1,"r2");
nameof(basering); basering;
↦ r2
↦ //   characteristic : 32003
↦ //   1 parameter    : a
↦ //   minpoly        : (a2+1)
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
```

```
factorize(y3-y+2,1);
↦ _[1]=y3-y+2
list L = splitring(y3-y+2,"r3",a);
↦ // new minimal polynomial: a6+241a4+4a3-12319a2-976a+597
r3;³
↦ //   characteristic : 32003
↦ //   1 parameter    : a
↦ //   minpoly        : (a6+241a4+4a3-12319a2-976a+597)
↦ //   number of vars : 3
↦ //        block   1 : ordering dp
↦ //                  : names    x y z
↦ //        block   2 : ordering C
L;
↦ [1]:
↦    (-6367a5+4077a4+9016a3+12428a2+1256a+2755)
factorize(y3-y+2,1);
↦ _[1]=y+(-1862a5-15302a4-8924a3+6601a2+10257a-13838)
↦ _[2]=y+(-6703a5-4690a4+14865a3-15843a2-11305a+7208)
↦ _[3]=y+(8565a5-12011a4-5941a3+9242a2+1048a+6630)
(x+L[1])*(x-L[1]);
↦ x2+1
factorize(z2-az+5,1);
↦ _[1]=z2+(-a)*z+5
list l = splitring(z2-az+5,"",list(L[1],z2-az+5));
↦ // new minimal polynomial: a12+271a10+4a9-7124a8-916a7+12066a6
     -4580a5+13918a4+500a3+9360a2+15625
nameof(basering);
↦ r3
l;
↦ [1]:
↦    (10516a11+9155a10-6937a9+6190a8+14420a7+5850a6+14553a5-2162a4+4675a3
      -11305a2-5884a+2882)
↦ [2]:
↦    z2+(850a11+6329a9+3400a8-6833a7-10528a6+15140a5+11366a4-10810a3
        +8961a2-12748a)*z+5
l[1]^2;
↦ -1
factorize(l[2],1);
↦ _[1]=z+(850a11+6329a9+3400a8-6833a7-10528a6+15140a5+11366a4-10810a3+8961a2-12747a)
↦ _[2]=z+(-a)
```

[3] the minimal polynomial of `r3` is based on random data

A.2.2 Documentation of the Procedures

```
LIBRARY:    primitiv.lib    PROCEDURES FOR FINDING A PRIMITIVE ELEMENT

primitive(ideal i);    find minimal polynomial for a primitive element
primitive_extra(i);    find primitive element for two generators
splitring(f,R[,L]);    define ring extension with name R and switch to it
```

For examples to the procedures look at the preceding example sessions or use the online help (type "`LIB "primitiv.lib"; example` <procedure>`;`" in Singular).

Primitive element of finitely many elements: `primitive`

USAGE: `primitive(i);` `i ideal` of the following form:

Let k be the ground field of the basering, $\alpha_1, \ldots, \alpha_n$ algebraic over k, $m_1(x_1)$, $m_2(x_1, x_2)$, $\ldots$, $m_n(x_1, ..., x_n)$ polynomials over k such that $m_j(\alpha_1, \ldots, \alpha_{j-1}, x) \in k(\alpha_1, \ldots, \alpha_{j-1})[x]$ is a minimal polynomial for $\alpha_j \in \overline{k(\alpha_1, \ldots, \alpha_{j-1})}$ for all $j = 1, \ldots, n$.

Then $m_1, \ldots, m_n$ can be chosen as the generators of `i`.

RETURN: `ideal` $\mathtt{j} \subset k[x_n]$ such that the generators fulfill

$\mathtt{j[1]} \in k[x_n]$ is a minimal polynomial for a primitive element β of $k(\alpha_1, \ldots, \alpha_n) = k(\beta)$

$\mathtt{j[2]}, \ldots, \mathtt{j[}n+1\mathtt{]} \in k[x_n]$ are polynomials s. t. $\mathtt{j[}i+1\mathtt{]}(\beta) = \alpha_i \quad \forall\, i = 1, \ldots, n$

NOTE: The number of variables in the basering has to be exactly the number n of given algebraic elements α_j (and minimal polynomials m_j).

The first element being tried as primitive element is α_n. If α_n is not primitive, and if the ring characteristic is zero or a big prime number, `primitive` computes random linear combinations of α_i. So in this case two invocations with the same parameter `i` will in general produce different results (seed Singular's pseudo random generator to avoid this).

If k has only few elements it can be that no one of the linear combinations of $\alpha_1, \ldots, \alpha_n$ is a primitive element. In this case `primitive` returns the zero ideal. If this occurs use the procedure `primitive_extra` instead.

Primitive element of two elements: `primitive_extra`

USAGE: `primitive_extra(i);` `ideal i=f,g;` with the following properties:

Let $k = \mathbb{Q}$ or $k = \mathbb{F}_p$ be the ground field of the basering $k[x, y]$, let α, β be algebraic over k, then: $\mathtt{f} \in k[x]$ is a minimal polynomial of α (over k), $\mathtt{g} \in k[x, y]$ has the property that $\mathtt{g}(\alpha, y)$ is a minimal polynomial of β (over $k(\alpha)$).

RETURN: `ideal` $\mathtt{j} \subset k[y]$ such that
$\mathtt{j[1]}$ is a minimal polynomial for a primitive element γ of $k(\alpha, \beta) = k(\gamma)$
$\mathtt{j[2]}$ is a polynomial s. t. $\mathtt{j[2]}(\gamma) = \alpha$

NOTE: While `primitive` may fail for finite fields, this procedure tries all elements of $k(\alpha,\beta)$ and hence finds by assurance a primitive element. In order to do this (to try all elements) field extensions like $\mathbb{F}_p(a)$ are not allowed for the ground field k.

On the other hand—unlike `primitive`—does `primitive_extra` not compute a representation of β by γ.

`primitive_extra` assumes that `g` is monic as polynomial in $(k[x])[y]$.

Algebraic ring extensions: `splitring`

USAGE: `splitring(f,R[,L]);` `f poly`, univariate and irreducible(!) over the active basering; `R string`, `L list` of `poly`s and/or `ideal`s (optional)

CREATE: `splitring` defines an algebraic ring extension of the current basering with name `R`, in which `f` is reducible[4] (`f` "splits"), and changes to it.

If the old ring has no parameter, the name "`a`" is chosen as the parameter of `R` (if `a` is not already a variable; if it is, the procedure tries some other names like "`b`" etc.; if it cannot find a free name, it produces an error message). Otherwise the name of the parameter is kept and only the minimal polynomial is changed. The names of variables and orderings are not affected.

NOTE: It is also allowed to call `splitring` with `R = ""`. Then the old basering will be *replaced* by the new ring (with the same name as the old ring).

RETURN: `list L` mapped into the new ring `R`, if `L` is given; else nothing.

ASSUME: The active ring must allow an algebraic extension (e. g. it cannot be a transcendent ring extension of $\mathbb{Q}$ or $\mathbb{F}_p$).

Keep in mind—if the current basering is already an algebraic extension of $\mathbb{Q}$ or $\mathbb{F}_p$—that the meaning of the parameter will be changed by `splitring`. Hence the inclusion mapping into the new ring is not trivial (do *not* use `imap`). You can solve this by collecting the ring dependent variables you want to use in the new ring (at least the data depending on the parameter) in a list and by passing this list as optional parameter to `splitring`. Also note that the procedure calls `primitive` and hence two identical invocations of `splitring` may generate different rings.

`splitring` shows the new value of the minimal polynomial—if the previous ring also had one—to remind the user of the different meaning of the ring parameter, and to make it easier to see if the choice of the new ring differs from a previous example due to randomness. This can be turned off by setting the Singular variable `printlevel` to a negative value.

[4] linear polynomials `f` should also work, though this will only lead to a different (in general more complicated) definition of the same ring and is not the aim of the procedure. A linear polynomial remains of course irreducible.

Literaturverzeichnis

[Ab] Abhyankar, S. S.: *Inversion and Invariance of Characteristic Pairs.* Am. Journ. of Math. 89 (1967), pp. 363–372

[BW] Becker, T.; Weispfennig, V.: *Gröbner Bases—A Computational Approach to Commutative Algebra.* GTM **141**, Springer, New York – Berlin – Heidelberg 1993

[BK] Brieskorn, E.; Knörrer, H.: *Algebraische Kurven.* Mathemat. Institut zu Bonn, 1978

[Ca] Campillo, A.: *Algebroid Curves in Positive Characteristic.* Lecture Notes in Math. **813**, Springer, Berlin – Heidelberg – New York 1980

[Coh] Cohen, I. S.: *On the Structure and Ideal Theory of Complete Local Rings.* Trans. Amer. Math. Soc. 59 (1946), pp. 54–106

[Cox] Cox, D.; Little, J.; O'Shea, D.: *Ideals, Varieties, and Algorithms.* 2nd ed., Springer, New York – Berlin – Heidelberg 1997

[Ei] Eisenbud, D.: *Commutative Algebra with a View Toward Algebraic Geometry.* GTM **150**, Springer, New York – Berlin – Heidelberg 1995

[Fi] Fink, J.: *Puiseux-Algorithmus und topologische Invarianten von Kurvensingularitäten.* Diplomarbeit, Uni Kaiserslautern 1993

[Go] Gorenstein, D.: *An Arithmetic Theory of Adjoint Plane Curves.* Trans. Amer. Math. Soc. 72 (1952), pp. 414–436

[GPS] Greuel, G.-M.; Pfister, G.; Schönemann, H.: *Singular version 1.2 User Manual.* Reports On Computer Algebra **21**, Zentrum für Computeralgebra, Uni Kaiserslautern 1998

[Ha] Hartshorne, R.: *Algebraic Geometry.* GTM **52**, Springer, Berlin – Heidelberg – New York 1977

[Ko] Koblitz, N.: *A Course in Number Theory and Cryptography.* GTM **114**, Springer, Berlin – Heidelberg – New York 1988

[Kru] Krull, W.: *Dimensionstheorie in Stellenringen.* Journ. für die Reine und Angewandte Math. 179 (1938), S. 204–226

[Ku] Kunz, E.: *Ebene algebraische Kurven.* Der Regensburger Trichter **23**, Uni Regensburg, 1991

[Mi] Milnor, J.: *Singular Points of Complex Hypersurfaces.* Annals of Math. Stud. **61**, Princeton University Press, 1968

[Ry] Rybowicz, M.: *Sur le calcul des places et des anneaux d'entiers d'un corps de fonctions algébriques.* Thèse d'Etat Université de Limoges 1990

[Wae] van der Waerden, B. L.: *Algebra I.* 8. Aufl., Springer, Berlin – Heidelberg – New York 1971

[Wa] Walker, R. J.: *Algebraic Curves.* Springer, New York – Heidelberg – Berlin 1978

[Za1] Zariski, O.: *Studies in Equisingularity III.* Am. Journ. of Math. 90 (1968), pp. 961–1023

[Za2] Zariski, O.: *Le problème des modules pour les branches planes.* Cours donné au Centre de Mathématiques de l'Ecole Polytechnique, Paris 1973

[ZaSa] Zariski, O.; Samuel, P.: *Commutative Algebra.* GTM **29**, Springer, Berlin – Heidelberg – New York 1960

Index

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe.