Bachelor's Thesis in Computer Science

Automated Generation of Kempe Linkages for Algebraic Curves in a Dynamic Geometry System

submitted by

Alexander Kobel

on September 26, 2008

Saarland University, Saarbrücken, Germany
Faculty of Natural Sciences and Technology I
Department of Computer Science

Supervisor and First Reviewer

Prof. Dr. Frank-Olaf Schreyer

Advisor and Second Reviewer

Dr. Oliver Labs

Non-plagiarism statement
Hereby I confirm that this thesis is my own work and that I have documented all sources used.

Statement of Consistence
Hereby I confirm that this thesis is identical to the digital version submitted at the same time.

Declaration of Consent
Herewith I agree that my thesis will be made available through the library of the Computer Science Department.

Saarbrücken, September 26, 2008

Alexander Kobel

Abstract

In 1876, Alfred Bray Kempe stated a preliminary version of what nowadays is known as *Kempe's Universality Theorem*: For any intersection of an algebraic curve with a closed disc in the Euclidean real plane $\mathbb{R}^2$, there is a linkage translating a motion along these curve segments to a straight line segment and *vice-versa*. Using linkages like the *Peaucellier inversor*, which constrain a hinge to a straight line segment, it is thus possible to piecewise delineate arbitrary algebraic curves. This is particularly remarkable in combination with the Stone-Weierstraß approximation theorem, which states that arbitrary continuous functions or curves inside a compact set can be interpolated in any precision by polynomial functions or algebraic curves.

In this thesis, we present the constructions involved in assembling those linkages, following Kempe's original approach and a more recent reformulation by Gao et al. In addition, we shortly outline some recent results by Abbott et al. on generalizations of Kempe's Universality Theorem to arbitrary dimensions.

The writing comes along with an implementation of a one-way simulation of the linkage design in a dynamical geometry system called *Cinderella*, which features so far unrivaled mathematical background for ruler-compass constructions without specialization to particular applications. "One-way" here refers to the translation from a movement on the curve to the straight line.

Information on the shape of the algebraic curve is retrieved via an Internet interface from the ongoing project *Xalci*, provided by the Algorithms and Complexity working group of the Max-Planck Institute for Computer Science. Their work aims on the topological analysis and visualization of implicit algebraic curves while providing guaranteed exactness of the results, and is to be extended for algebraic surfaces and curves in higher dimensions.

Our simulation tries to give as much freedom to the user as possible, within the limits set by computational complexity of a theoretically perfectly general approach. As of the publishing of this thesis in the end of September 2008, the interface is publicly available at `http://www.math.uni-sb.de/ag/schreyer/Kempe-Linkage/`.

Acknowledgements

Behind the scenes of this thesis there are a lot of patient and helpful people, who have spent time and work in my favor. I want to mention the most important ones here and thank them for their support, not only while I was writing this thesis, but during my whole studies.

The first to mention is *Timo von Oertzen*, who led me on my first steps of the way into algebraic geometry—probably without even knowing about this.

At this time, he was employed as an assistant of *Frank-Olaf Schreyer*, who assured me to again care about maths, motivated me by giving interesting lectures, and eventually agreed to supervise me for the writing of this thesis.

The topic of the latter was proposed by Timo‘s successor *Oliver Labs*. Not only my advisor and the person to speak to when you have issues with anything in geometry, he also knows the right persons to call for help: *Ulrich Kortenkamp*, the main author of *Cinderella*, and *Pavel Emeliyanenko* and the *Algorithms and Complexity working group* at the *Max-Planck-Institute for Computer Science*, without whom the accompanying practical part of this thesis would not have been possible to do.

Besides, Oliver also is someone you want to have a chat with for a cup of coffee. This eventually leads to the *Teestube* of the departement of mathematics in Saarbrücken, where—despite it‘s name—you have a hard time looking for tea, but you always get coffee. Lately this is *Daniel Fries* merit, being the donator of a Senseo coffee maker. Along with *Andreas Fromkorth* and *Tobias Schnur*, he is one of the many “inhabitants” of the Teestube to mention here for being the partners of hours of fruitful discussions, although not always on mathematics, who so helped me to find my way to and through this thesis.

Contents

List of Figures

1 Introduction

1.1 Kempe Linkages

A fundamental task of engineering is the problem of assembling mechanisms which make a distinct element move along a certain path. While recent achievements in electrical engineering allow positioning of devices with incredible precision by means of microcontrollers, larger scale applications still rely on classical solutions: gears using shafts, chains, belts or ball bearings can be found in almost every imaginable mechanical part, from wind engines down to the fans in your laptop. Although the construction of these parts clearly involves difficulties, like natural restrictions on the size, the robustness and degree of efficiency is often the telling argument for the use of those classical components.

Probably the best known example is James Watt's "parallel motion" linkage, dating back to 1784 and employed in early locomotives operated by steam engines. It is used to convert the linear motion of the piston to drive the rotation of the wheels. While the parallel motion almost perfectly answers it's purpose, it is far from a theoretically perfect solution on the translation of a straight line to a circle.

In 1876, Alfred Bray Kempe used the components of Watt's engine to develop linkages to piecewise delineate arbitrary algebraic curves, starting from their implicit polynomial form. Here, a linkage is—like Watt's parallel motion—a device assembled of rigid bars of fixed length, connected by hinges on their endpoints. The linkwork translates to a crank describing a circle, or a straight line, which is proven to be equivalent by Charles-Nicolas Peaucellier's invention of the *Peaucellier cell* which, in contrast to Watt's linkage, is capable of *exactly* converting rotations to linear motions.

Probably the most elegant reformulation of *Kempe's Universality Theorem*, how it is referred to these days, originates from William Thurston. By applying the Stone-Weierstraß approximation theorem, he follows that there is a series of linkworks approximating any continuous bounded plane curve. Thurston therefore summarized that "there is a linkage that signs your name", assuming you have a finite name.

Kempe slightly criticized Euclid for his stressing of ruler-compass construction without actually giving a hint how to build an ideal ruler in the first place. Thus, it appears ironical that Kempe himself is best known among mathematicians for his false proof of the four colour theorem, and he also turned out to be careless in his linkage design. However, both these works rendered a great service to the mathematical community and have been corrected later; both proofs involve Kempe‘s original key ideas.

With his work, Kempe eventually settled an important question in engineering from the theoretical point of view; although he gave a constructive description of the design of the linkwork, the complexity of the mechanisms exceeds the practically relevant limit. It remained an open field to the engineers to figure out easier approximate solutions; however, simulations of the linkages turned out to be useful in modern applications, such as CAD and robotics.

Kempe‘s conclusions have been extended later to arbitrary dimensions. With some additional restrictions not considered by him, there still is uncharted territory left to further research.

1.2 Dynamic Geometry in Cinderella

Everyone of us may with mixed feelings remember the geometry lessons in school. Probably nobody dealt with the class without some torn sheets of paper, with drawings at the wrong scale or a cluttered choice of parameters for constructions.

Dynamic geometry systems (DGS) tackle this problem, allowing to arbitrarily zoom and shift the viewport or modify some input elements and automatically rearrange the depending parts accordingly. The current calculating capacity of usual personal computers, available at cheap prices and in increasing quantities found in schools, also allow interactive animations and tracing of elements under movements of others, capable of drastically easing the teaching of the relation of mathematics and the "real world". Beyond the applications in high school lessons, dynamic geometry programs increasingly aim to include more in-depth topics, like introductory projective geometry, which demand a great deal of imagination from the student.

Amongst a number of tools mainly targeted at the profitable market of high school pupils, *Cinderella* takes a special position. Throughout the twelve-years developement process of *Cinderella* up to now, the authors Ulrich Kortenkamp and Jürgen Richter-Gebert took great care of implementing a thorough mathematical model of geometry.

Besides features such as polar views on constructions or the representation of hyperbolic geometry, there are two major differences in design compared to standard systems: First, all elements of a *Cinderella* construction are considered in the complex projective plane, which guarantees consistency of constructions even in degenerate positions. Second, *Cinderella* strictly adheres the concept of continuous movements. Both features are—in this extent—only competed by a discontinued project named "pdb" ("projective drawing board") by Harald Winroth, whose workings on dynamic geometry theory the authors of *Cinderella* base on. One benefit of this approach is that, up to now, *Cinderella* is the only dynamic geometry system able to draw complete loci, i.e. traces of objects under continuous movements of others.

An additional advantage of *Cinderella* for more complex constructions is it's embedded script editor, which grants the full power of a high-level programming language with focus on it's geometric operations. Hereby, it spares the user from the necessary calculations performed in background—it suffices to deal with the constructions semantics.

Finally, *Cinderella* offers the possibility to work embedded in web pages as a Java applet, which renders a great opportunity for proof-of-concept presentations.

These outstanding capabilities as well as Kortenkamp's offer to support our project are our reasons to choose *Cinderella* as the basis for an implementation of Kempe's algorithm in a modern geometry environment.

1.3 Algebraic Geometry Visualization in Xalci

Rendering of algebraic curves and surfaces are *the* main underlying procedures of almost every recent geometric modeling application. However this is typically done within tight boundaries on the complexity of the mathematical object. Font rendering, standard vector drawing tools or 3D modeling and computer aided design (CAD) usually employ at most nonuniform rational B-spline surfaces (NURBS), which nowadays can be animated in real-time without major trouble for reasonable input sizes.

Still there is a need for handling higher degree objects without a given parametrization, respecting exactness guarantees on both shape and topology. For example, industrial tools for computer aided manufacturing in specialized high-precision applications have to provide seemless transitions which exceed the possibilities of spline approximations.

This is the long-term goal of *Xalci*, part of the *EXACUS* project for Exact Algorithms on Curves and Surfaces at the Algorithms and Complexity working group of the Max-Planck-Institute for Computer Science in Saarbrücken. *Xalci* treats the topological correct analysis of arrangements of implicit algebraic curves, to be extended to also cover space curves and surfaces. Subsequently, simpler subdivisions with easy topology are each rasterized independently, giving an exact visualization of the curve up to output resolution of the images.

While the algorithms or stand-alone implementations are not publicly available (yet), the results can be seen via a Flash web interface at `http://exacus.mpi-inf.mpg.de/cgi-bin/xalci.cgi`. This motivated the idea of utilizing the exisiting web transport to gather the shape information of curves needed for rendering and animation of the Kempe mechanisms, kindly supported by the *Xalci* team, in particular represented by Pavel Emeliyanenko.

Together with *Cinderella*'s web presentation features, we can provide a comprehensive simulation of the line-curve-translation through a easily usable web interface.

2 Basic Linkages

In this chapter, we present basic linkages, which will be our elementary toolkit for the more complicated constructions to follow. We require the reader to be familiar with usual well-known results from elementary geometry; a basic knowledge of algebraic geometry is assumed throughout the following chapters.

In particular, in section 2.1 we discuss early attempts on describing a straight line by means of a linkage. Our goal is the presentation of the Peaucellier-Lipkin cell, a linkage which solves this problem exactly. Afterwards, we will express translation of lengths (2.2) as well as rotation (2.3), and addition and multiplication of angles (2.4) in terms of mechanics.

The linkworks given in this chapter are taken from [GZCG02], and are chosen with theoretical aspects in mind. Especially the number of links—i.e. the combinatorial complexity of the mechanisms—is stressed. For practical applications, a number of other factors have to be considered, for example the smoothness of movements or the stability of the construction.

For this reason, the linkages presented here are by no means unique. However, for some akin constructions of similar simplicity, Kempe writes: [Kem77]

> In this form, which is a very compact one, the motion has been applied in a beautiful manner to the air engines which are employed to ventilate the Houses of Parliament. The ease of working and absence of friction and noise is very remarkable.

2.1 Straight Line Motion

2.1.1 How to Draw a Straight Line?

The first problem we encounter in finding linkages for algebraic curves seems to be the most simple case possible: How can we describe a straight line? In fact, this very problem is what inspired Kempe to give a lecture about linkages in 1877. In the lecture notes ([Kem77]) he states

> But the straight line, how are we going to describe that? Euclid defines it as "lying evenly between its extreme points." This does not help us much. Our text-books say that the first and second Postulates postulate a ruler. But surely that is begging the question. If we are to draw a straight line with a ruler, the ruler must itself have a straight edge; and how are we going to make the edge straight? We come back to our starting point.

To the best of our knowledge, it is not clear whether Euclid itself recognized this problem.

For start, we denote that—given a perfect plane—a circle is straightforward to design by linkages. We pick some center point on the plane and stick a pivot there; on this pivot, we attach any kind of rigid material. This allows a circular motion of the shape around the bolt. We now choose some arbitrary point on the shape; tracing this point while rotating the shape yields, from the theoretical point of view, a perfect circle.

Now the reader might ask where the difference lies in the imagination of a perfect ruler and this construction, involving "a perfect plane" and infinitely small "points" and pivots. For our linkage, we are able to adopt the precision according to our needs. In particular, we do not demand the existence of a sample piece of the result. Instead, we rely on basic elements whose overall influence on the relative error of the result vanishes proportional to the size of the linkage—i.e., the radius of the circle. In a perfect scenario both the pivot and it's bearing are circular; however, this is not due to the shape of the curve we want to trace, but a basic requirement for *any* link of an arbitrary mechanism to achieve smooth transitions.

If we do not have high-precision components for our drawing, we might just take some branch as pivot and a tensed rope—which essentially fulfills the same requirements as a rigid shape—and draw a circle in a larger scale, achieving a "roundness" that may well compete today's industrial parts.

It remains to note that the basic plane is a real restriction; on the other hand, the same holds for all ruler-compass constructions, so we might have to accept it for granted.

2.1.2 An Inexact Approach: Watt's linkage

Despite it's seemingly easy nature and high impact on technicial engineering, the straight-line problem remained unsolved for long time. The best known approach, which was directly influenced by the needs of upcoming technology—namely locomotives employing a the steam engine—is James Watt's Parralel Motion, dating back to 1784. Despite it's name, Watt did not succeed in generating *parallel* curves; in fact,

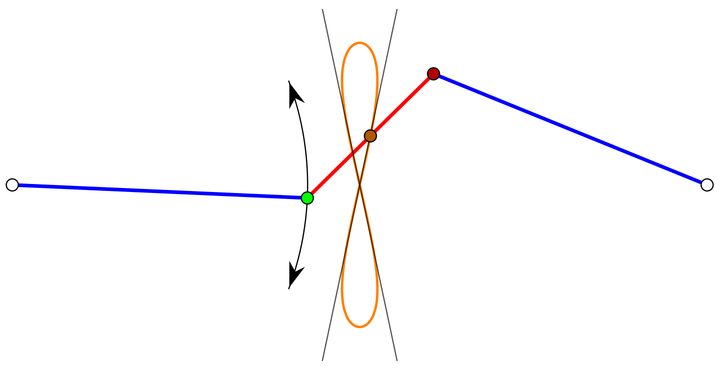


Figure 2.1: Watt's linkage

he could not even solve the straight-line problem. However, for the sake of historical background of linkages, we will shortly explain Watt's linkage and the resulting curve.

The linkage (figure 2.1) consists of three bars, consecutively connected by joints. The outer bars share the same length and are each fixed by pivots on the underlying plane.

Now, we trace the center point of the inner bar under rotation of any of the outer bars. (Of course, it does not matter which one we move, since each determines the position of the other throughout the motion.)

The resulting curve, called Watt's lemniscate, is an algebraic curve of degree six. From a mean position, i.e. a situation where the pencil exactly coincides with the center of the pivots, the linkage approximates a straight line. The farther we go from the mean, the larger gets the deviance from the line; after a full rotation, we end up with a closed curve. We can thus state that Watt's linkage does not allow any configuration to describe a straight line, because by Bézout's theorem a component of an algebraic curve cannot both contain a straight line and at the same time be closed in the real plane $\mathbb{R}^2$.

2.1.3 An Exact Solution: The Peaucellier-Lipkin Cell

The first exact solution to the straight-line problem by a planar linkage was given in 1864 by Charles-Nicolas Peaucellier. His invention was not recognized by the scientific community in the first place, until Lippman Linkin rediscovered and published his work.

The *Peaucellier-Lipkin linkage* (or just *Peaucellier cell*) consists of seven bars in a configuration as shown in figure 2.2. Here, $|AB| = |BC| = |CD| = |DA| = a$, $|PB| = |PD| = b$, and $|OP| = |OA| = c$. Furthermore, let E be the intersection point of the diagonals AC and BD of the rhombus $ABCD$.

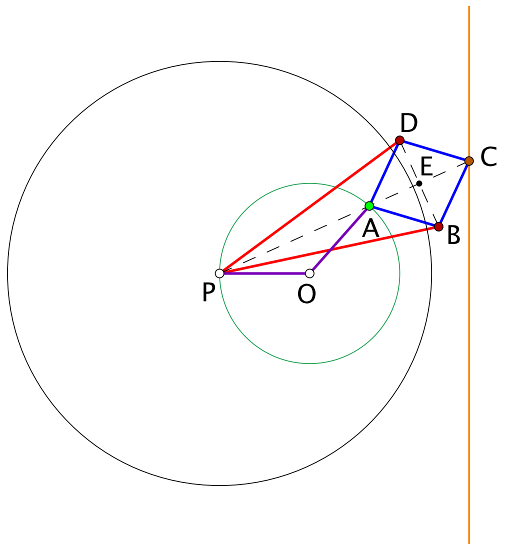


Figure 2.2: The Peaucellier-Lipkin cell

It holds

$$\begin{aligned} |PA| \cdot |PC| &= (|PE| - |AE|) \cdot (|PE| + |AE|) \\ &= |PE|^2 - |AE|^2 \\ &= b^2 - |BE|^2 - |AE|^2 \\ &= b^2 - a^2 \\ &= const \end{aligned}$$

Thus, C is the inversion point of A with respect to some circle with center P. The inverse of any circle through P with respect to this circle is a line; in particular, this holds for the circle centered in O. Therefore, when A rotates around the pivot O, the trace of C is a straight line perpendicular to PO.

Due to the limited length of the links, the locus of C under all possible configurations of the linkwork is not a complete line, but a line segment, called a *slot*. This amounts to performing the necessary calculations in the real numbers. If we consider an algebraically closed field, the resulting curve actually *is* a line, i.e. the vanishing locus of a two-variate polynomial of degree 1.

In particular, in the complex numbers $\mathbb{C}$, where the *Cinderella*'s computations take place, the hinges B and D eventually leave the real plane. However, the pencil C stays on a real line. Details as well as a similar example are given in section 4.1.1.

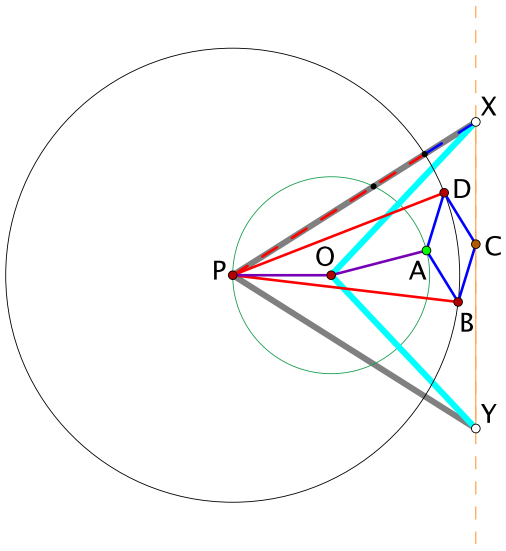


Figure 2.3: The Peaucellier cell on slot

2.1.4 The Peaucellier Cell on Slot

We can exploit this limitation of movement to constrain a point in a slot. Starting from two distinct points X and Y, we attach bars of pairwise equal length to get O and P.

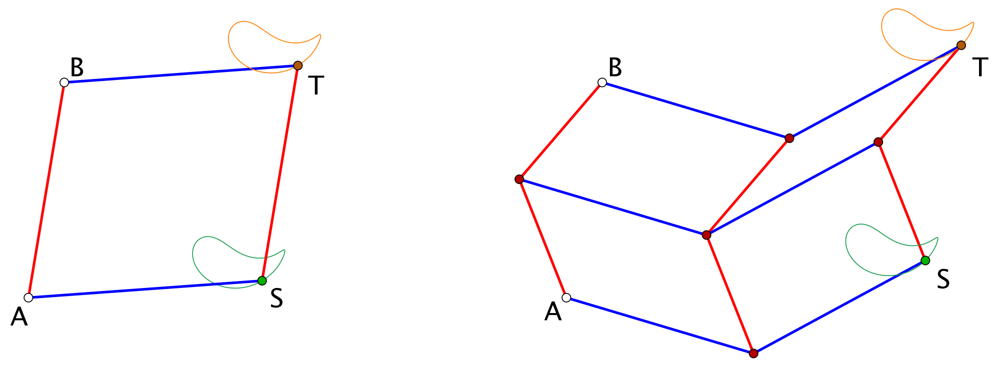
The extremal position of the Peaucellier cell occurs if and only if B and D coincide and, thus, the bars AB and BC and likewise AD and DC are on the same line XP (or YP). Accordingly, we choose $a = \frac{|XP|-c}{2}$ and $b = |XP| - a$ to get a linkage tracing out the segment XY.

Note that we do not rely an any point on XY but the pivots itself; therefore, we are able to specify the slot without a template.

From now, when we say that a point is *in a slot*, it means that we use this very modified Peaucellier linkage to do so. The slot itself still can be moved freely along with the attached linkwork; we say that the slot is *on a platform* determined by X and Y.

2.2 The Translator

The next tool to be used in linkage design translates a distance to some other place. The most basic idea is to build a parallelogram, in which the bars AB and ST and AS and BT pairwise share the same length. If we attach A and B to some pivots, moving S in the plane accordingly moves T; ST is a translation of AB.



(a) (b)

Figure 2.4: Two translators.

Note that the length of the bars is both a necessary and sufficient condition for the shape of the linkwork, as long as we guarantee that no crossings of the bars occur, leading to an antiparallelogram.

While this simple mechanism works flawlessly in theory, it is not convenient, since the distances $|AB|$ and $|AS|$ are fixed. We can do better with a combination of four such translators (see figure 2.4 b). The correctness of this linkage directly results from the fact that parallelity is an equivalence relation of lines in the plane and, as such, in particular transitive.

For the sake of simplicity, we can assume that any parallel bars have the same length. This assumption is without loss of generality; the only real restriction is given by the sum of the lengths of the bars compared to $|AB|$ and $|AS|$.

When in the following we say that a segment or distance is *translated* to another point, we refer to the use of this refined linkage to achieve this.

2.3 The Rotator

To complete the movements of segments in the plane, in addition to translation we have to be able to perform rotations. Figure 2.5 shows a linkwork to achieve this.

Here, $|OA| = |OB|$, $|AC| = |BC|$, $|DS| = |DT|$, and $|DU| = |DV|$. $|OA| = |OB|$ is not a real restriction; if source and target direction (along OA or OB) are given with

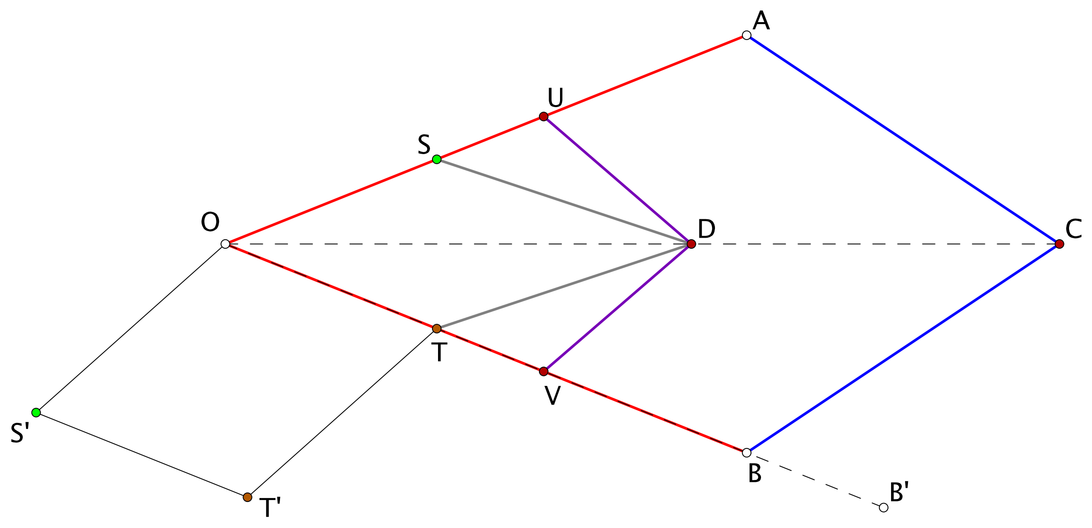


Figure 2.5: The distance copier

distinct lengths, say $|OB'| > |OA|$, we use a platform to constrain B in the slot OB'. Attaching the rotator to the pencil of the Peaucellier linkage accordingly forces a unique position of B where $|OA| = |OB|$.

In the same manner, S, U, T and V are constrained in OA or OB by platforms. Again, the positions of U, V and, thus, also T are uniquely determined by the position of S in it's slot.

Now we observe that the construction formed by those restrictions consists of three kites $OSDT$, $OUDV$ and $OACB$, which share their axis of symmetry. In particular, OS and OT are the images of each other under mirroring along OC, yielding $|OS| = |OT|$.

2.3.1 The Distance Copier

Attaching a translator to OT extends this rotator to the *distance copier*, which assures that $S'T'$ arises as the image of OS under an Euclidean move.

Note that, although the translator is sketched but with a parallelogram, we need the more complex translator (figure 2.4 b) in this situation: in general, we do not know $|OS|$ in advance and thus need the flexibility of distances granted by this translator, not only to be able to move the linkage, but even to assemble it in a position where all links fit in the first place.

2.4 Angle Adder and Multiplicator

We finish the presentation of the basic linkworks with two constructions on angle operations. In particular, we show that linkage designs include all necessary tools to implement the $\mathbb{Z}$-module structure of angles.

2.4.1 The Angle Adder

A common task in linkage design is the addition of a (finite) number of angles. This can easily be achieved by inductively adding two angles by means of an *angle adder* consisting of distance copiers:

Suppose we are given two angles $\measuredangle AOB$ and $\measuredangle CO'D$. W.l.o.g. we can assume $|OA| = |OB| = |O'C| = |O'D|$; otherwise, we use platforms to constrain the links accordingly. We then attach two distance copiers to move $CO'D$ such that the image of $O'C$ is OA, yielding $\measuredangle CO'D = \measuredangle C'O''D' = \measuredangle AOD'$, and now use another distance copier to move $|AB|$ to $D'T$ such that AB is rotated by the angle $\measuredangle AOD' = \measuredangle CO'D$.

Since we demanded the lengths of the legs to be equal, both AB and AD' are chords in a circle with center O. Therefore, $\measuredangle AOB + \measuredangle CO'D = \measuredangle AOB + \measuredangle AOD' = \measuredangle AOT$.

2.4.2 The Angle Multiplicator

Now suppose we want to construct an *angle multiplicator*, i.e. a mechanism to multiply an angle by an integer. Again, we can assume both legs to share the same lengths, so we can use a distance copier in the same manner to double, triple, ... the angle.

However, there is an easier approach to solve this special case, shown in figure 2.6. Here $OADB$ and $OBEC$ are antiparallelograms; the links satisfy $|OA| \cdot |OC| = |OB|^2 \Leftrightarrow \frac{|OA|}{|OB|} = \frac{|OB|}{|OC|}$.

Of course we may *not* allow $|OA| = |OB|$ in this setting, which leads to the degenerate position $D = 0$. This again is not a real restriction, since we can employ platforms to get distinct lengths of the legs. In general, we will have to do so anyway, because we have to guarantee the ratio of the bar lengths.

Since $\frac{|OA|}{|OB|} = \frac{|OB|}{|OC|}$, $OADB$ is similar to $OBEC$, yielding $\measuredangle AOB = \measuredangle BOC$ and thus $\measuredangle AOC = \measuredangle AOB + \measuredangle BOC = 2\measuredangle AOB$. Iterating the construction, we can get a n times angle multiplicator for arbitrary, but fixed $n \in \mathbb{N}$.

Kempe [Kem76] uses the term *angle reversor* to call the multiplicator, because $\measuredangle BOA = -\measuredangle COB$; thus the mechanism also allows to change the orientation of arbitrary angles. Together with the angle adder, we are able to subtract angles. As a side remark, which

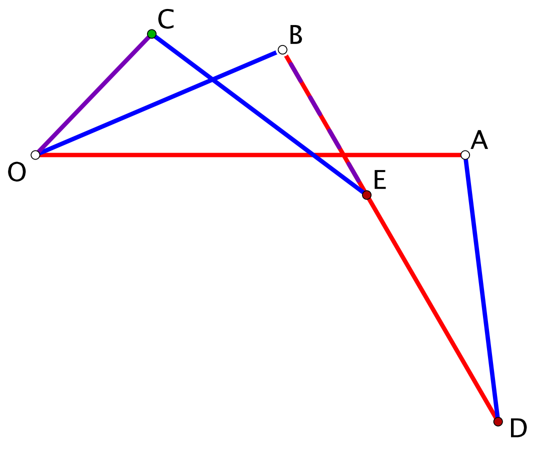


Figure 2.6: The angle multiplicator

is not necessary for our work, we further note that the multiplicator also allows to bisect angles by attaching O, A and C to the relevant points; then, $\measuredangle AOB = \frac{1}{2}\measuredangle AOC$.

3 Kempe Linkages for Plane Curves

The linkworks presented in chapter 2 are our tools for expressing algebraic terms in mechanisms. We have already seen in the discussion of the Peaucellier inversor that the most basic operation in linkage design is rotation. This suggests that we do better not to describe algebraic curves in the Euclidean plane in Cartesian coordinates as usual, but merely use some variant of a polar coordinate system.

In this chapter we thus explain how to express polynomials and, thus, algebraic curves by means of trigonometric expressions. We will see that what Abbott et al. [ABD08] call "trigonometric algebra" allows to reduce all occuring terms to some simple canonical form. In particular, we find out how to cancel arbitrary occurrences of any variable using linkages attached to only two points, the origin and any chosen unit representing the x-axis.

This directly leads to a constructive proof for *Kempe's Universality Theorem*, which states that for every bounded part of a plane algebraic curve there is a linkwork that translates a motion along the curve to a motion on a straight line segment. In consequence, there exists a series of linkworks to delineate any given algebraic curve in any region of the plane.

We close the chapter with a discussion of the complexity of this linkwork, including some recent results of Abbott et al. [ABD08], who generalize Kempe's work to arbitrary dimensions, and shortly explain their criticism and corrections on Kempe's proof.

3.1 Trigonometric Algebra

Throughout this section and the following we give a proof of

Kempe's Universality Theorem. *Let $f \in \mathbb{R}[x,y]$ be a polynomial defining an algebraic curve $\mathcal{C} = \mathcal{V}(f) = \{(x,y) \in \mathbb{R}^2 : f(x,y) = 0\}$, and D be a closed disc in the plane.*

Then there exists a linkwork translating a finite motion of a point S along a straight line segment to a motion of a point P along $\mathcal{C} \cap D$ and vice-versa.

Note that this formulation does not imply a continuous or even connected movement

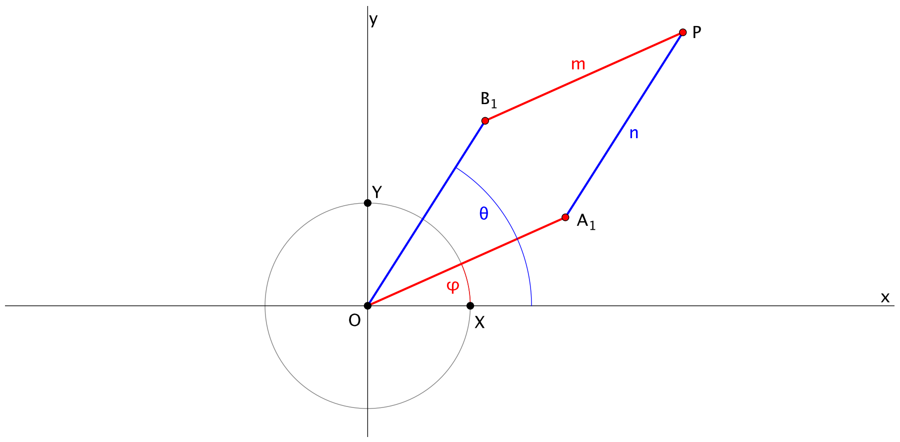


Figure 3.1: The parallelogram defined by P

of the linkage; we only demand that there is a single linkage s.t. a possible configuration of the bars, with S coinciding with the line segment, exists if and only if $P \in \mathcal{C} \cap D$.

For the proof, suppose we start with a given twovariate polynomial of total degree d in Cartesian coordinates

$$f(x,y) = \sum_{0 \le i+j \le d} a_{i,j} x^i y^j \quad \in \mathbb{R}[x,y], \tag{3.1}$$

implicitly describing the algebraic curve $\mathcal{C} = \mathcal{V}(f)$ of it's roots.

We demand the origin O and a unit X, determining the Cartesian x-axis, to be fixed. For any given point $P = (x,y) \in D$ in a certain distance to O we can attach a simple parallelogram linkage as shown in figure 3.1. The point P is free to move, constrained only by the lengths m and n of the brackets of the parallelogram; these will be determined later, but obviously have to be positive.

The bars of the parallelogram give angles φ and θ with the x-axis, so we can express P by

$$\begin{aligned} x &:= m\cos\varphi + n\cos\theta \\ y &:= m\sin\varphi + n\sin\theta. \end{aligned} \tag{3.2}$$

Substituting this into the polynomial of the curve (3.1) and plugging in the following identities

$$\sin\alpha = \cos\left(\alpha - \tfrac{\pi}{2}\right), \tag{3.3}$$

$$\cos^n\alpha = \frac{1}{2^n}\sum_{k=0}^{n}\binom{n}{k}\cos\left((n-2k)\alpha\right) \text{ and} \tag{3.4}$$

$$\cos\alpha\cos\beta = \frac{1}{2}\left(\cos(\alpha+\beta)+\cos(\alpha-\beta)\right) \tag{3.5}$$

yields

$$\begin{aligned}
f(x,y) &\overset{(3.2)}{=} \sum_{0\le i+j\le d} a_{i,j}\left(m\cos\varphi + n\cos\theta\right)^i\left(m\sin\varphi+n\sin\theta\right)^j\\
&\overset{(3.3)}{=} \sum_{0\le i+j\le d} a_{i,j}\left(m\cos\varphi + n\cos\theta\right)^i\left(m\cos\left(\varphi-\tfrac{\pi}{2}\right)+n\cos\left(\theta-\tfrac{\pi}{2}\right)\right)^j\\
&= \sum_{0\le i+j\le d} a_{i,j}\left(\sum_{k=0}^{i} c_{i,k}m^k n^{i-k}\cos^k\varphi\cos^{i-k}\theta\right)\cdot\\
&\qquad\cdot\left(\sum_{l=0}^{j} c_{j,l}m^l n^{j-l}\cos^l\left(\varphi-\tfrac{\pi}{2}\right)\cos^{j-l}\left(\theta-\tfrac{\pi}{2}\right)\right)\\
&= \sum_{0\le i+j\le d} a_{i,j}\sum_{k=0}^{i}\sum_{l=0}^{j} c_{i,k}c_{j,l}m^{k+l}n^{i+j-k-l}\cos^k\varphi\cos^{i-k}\theta\cos^l\left(\varphi-\tfrac{\pi}{2}\right)\cos^{j-l}\left(\theta-\tfrac{\pi}{2}\right)\\
&\overset{(3.4)}{\underset{(3.5)}{=}} \sum_{0\le s\le d}\ \sum_{-d\le t\le d}\left(\boldsymbol{a}_{s,t}\cos(s\varphi+t\theta)+\boldsymbol{b}_{s,t}\cos\left(s\varphi+t\varphi-\tfrac{\pi}{2}\right)\right).
\end{aligned}$$

For $s = t = 0$ we can isolate the constant terms to finally get

$$f(x,y) = \boldsymbol{c} + \sum_{\substack{0\le s\le d,\ -d\le t\le d\\(s,t)\ne(0,0)}}\left(\boldsymbol{a}_{s,t}\cos(s\varphi+t\theta)+\boldsymbol{b}_{s,t}\cos\left(s\varphi+t\varphi-\tfrac{\pi}{2}\right)\right) \tag{3.6}$$

where $\boldsymbol{a}_{s,t}, \boldsymbol{b}_{s,t}, \boldsymbol{c} \in \mathbb{R}$.

Gao et al. [GZCG02] point out that we can simplify this even further to get

$$f(x,y) = \boldsymbol{c} + \sum_{\substack{0\le s\le d,\ -d\le t\le d\\(s,t)\ne(0,0)}}\left(\boldsymbol{d}_{s,t}\cos(s\varphi+t\theta+\psi_{s,t})\right)$$

where $\boldsymbol{c}, \boldsymbol{d}_{s,t}, \psi_{s,t} \in \mathbb{R}$. This simplification however is, despite it's minor impact on the theoretical complexity of the construction, objectionable for our needs, since $\psi_{s,t}$ in general is not constructible by ruler-compass constructions even if $f(x,y) \in \mathbb{Q}[x,y]$.

For the theory this does not matter; here we can imagine arbitrary lengths. We will go into this topic in depth in chapter 4.

3.2 Kempe‘s Straight Line Linkwork

After all these transformations, we get a rephrased version of the defining equation of the algebraic curve, which depends on m, n, φ and θ instead of x and y. Since for an instance of a construction the bar lengths m and n of the parallelogram linkage attached to P stay fixed, we will from now on refer to the polynomial of equation (3.1) as $\tilde{f}_{m,n}(\varphi, \theta) := f(m \cos \varphi + n \cos \theta, m \sin \varphi + n \sin \theta)$.

Note that this is nothing but a translation into another coordinate system w.r.t. m and n; if and only if the angles φ and θ are chosen s.t. the corresponding point $P = (x, y)$ satisfies $f(x, y) = 0$, $\tilde{f}_{m,n}$ will equally vanish for those angles. Further, the coefficients $\boldsymbol{a}_{s,t}$, $\boldsymbol{b}_{s,t}$ and $\boldsymbol{c}$ do not depend on φ and θ, but are determined only by m and n.

We will now design a linkwork to achieve that for an arbitrary P on the curve $\mathcal{C}$, a distinguished hinge of the linkwork will be on a fixed straight line. We proceed as follows:

0. Let O be the origin of the coordinate system, X the unit point on the x-axis and $A_1, B_1 \notin \{O, P\}$ be the endpoints of the parallelogram OA_1PB_1 s.t. $\measuredangle XOA_1 = \varphi$ and $\measuredangle XOB_1 = \theta$.

1. First, we construct a static frame to get a link Y, corresponding to the unit on the y-axis. Here it is sufficient to put a isosceles triangle with bar lengths twice 1 and $\sqrt{2}$ on OX.[1]

2. For all integers s and $t \leq d$ occuring in (3.1), we attach angle multiplicators to the bars OA_1 and OB_1 to construct links OA_s, $s = 2, \ldots, u$ and OB_t, $t = -d, \ldots, -1, 2, \ldots, d$, satisfying $\measuredangle XOA_s = s\varphi$ and $\measuredangle XOB_t = t\theta$.

3. By means of angle adders, we construct links $OC_{s,t}$ from the A_s and B_t satisfying $\measuredangle XOC_{s,t} = \measuredangle XOA_s + \measuredangle XOB_t = s\varphi + t\theta$.

[1]While this leaves no degree of freedom on the shape of the triangle, we cannot avoid the congruent case, where we actually get $-Y$. This is a problem intrinsic to every construction of Y; to get a determined orientation, we have to also fix a third point in addition to O and X, which ideally is Y.

Furthermore, if we want to avoid the irrational ratio of lengths, we can use bar lengths corresponding to a Pythagorean triple to get the angle $\frac{\pi}{2}$ and use a distance copier to transmit the length OX on the y-axis.

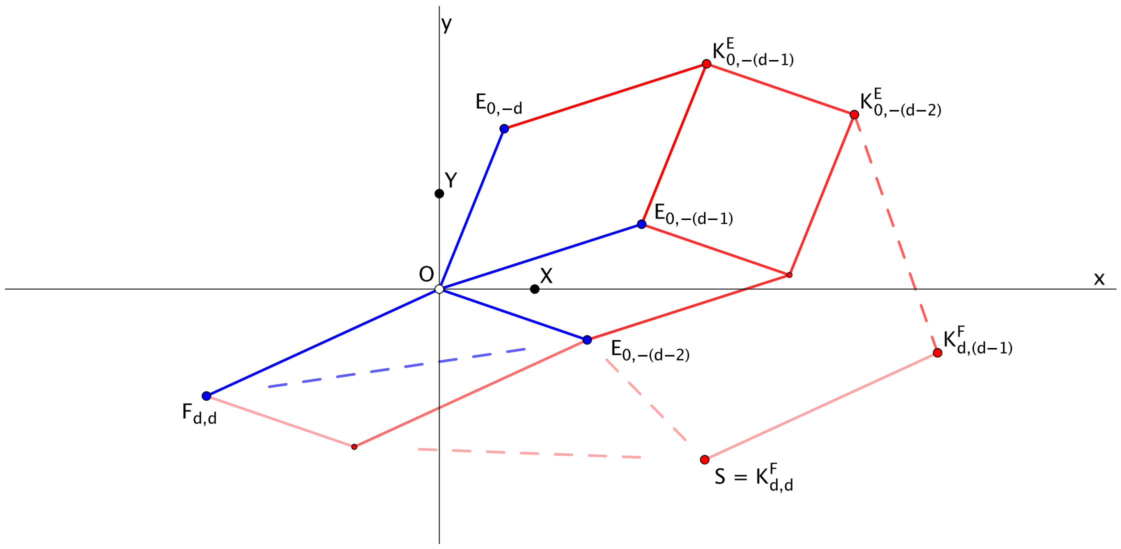


Figure 3.2: The final translations of Kempe's construction

4. In the same manner, we use angle adders on the $OC_{s,t}$ and OY to get points $D_{s,t}$ with $\measuredangle XOD_{s,t} = \measuredangle XOC_{s,t} - \measuredangle XOY = s\varphi + t\theta - \frac{\pi}{2}$.

5. For every coefficient $\boldsymbol{a}_{s,t}$ and $\boldsymbol{b}_{s,t}$ in (3.1), we attach a bar $OE_{s,t}$ or $OF_{s,t}$ of corresponding length to scale $OC_{s,t}$ or $OD_{s,t}$ s.t. $|OE_{s,t}| = \boldsymbol{a}_{s,t}$, $\measuredangle XOE_{s,t} = s\varphi + t\theta$ and $|OF_{s,t}| = \boldsymbol{b}_{s,t}$, $\measuredangle XOF_{s_t} = s\varphi + t\theta - \frac{\pi}{2}$.

6. Finally, we geometrically represent the summation by a chain of translators: We translate $OE_{0,-d}$ to $E_{0,-(d-1)}$ to get $K^E_{0,-(d-1)}$, $OK^E_{0,-(d-1)}$ to $OE_{0,-(d-2)}$ to get $K^E_{0,-(d-2)}, \ldots, OE_{1,-d}$ to $K^E_{0,d}$ to get $K^E_{1,-(d-1)}, \ldots,$ to get $K^E_{d,d}$. In the same manner, we add the $F_{s,t}$, to ultimately obtain $K^F_{d,d} =: S$.

Now, by construction, the links $E_{s,t}$ and $F_{s,t}$ have x-coordinate equal to $\boldsymbol{a}\cos(s\varphi + t\theta)$ and $\boldsymbol{b}\cos\left(s\varphi + t\theta - \frac{\pi}{2}\right)$. Therefore, the x-coordinate of S is

$$\begin{aligned} x &= \sum_{\substack{0\le s\le d,\ -d\le t\le d\\ (s,t)\neq(0,0)}} \left(\boldsymbol{a}_{s,t}\cos(s\varphi + t\theta) + \boldsymbol{b}_{s,t}\cos\left(s\varphi + t\varphi - \tfrac{\pi}{2}\right)\right) \\ &= \tilde{f}_{m,n}(\varphi,\theta) - \boldsymbol{c} \\ &= f(x,y) - \boldsymbol{c}. \end{aligned}$$

When P moves along $\mathcal{C}$, $f(P) = f(x,y) = \tilde{f}_{m,n}(\varphi,\theta) = 0$; accordingly, for S

$$x = f(x,y) - \boldsymbol{c} = -\boldsymbol{c}$$

holds. *Vice-versa*, if S moves along the line $\mathcal{V}(x+\boldsymbol{c})$, the locus of P is the curve $\mathcal{C}=\mathcal{V}(f)$.

This can be achieved by connecting S to a Peaucellier cell, constraining S to $\mathcal{V}(x+\boldsymbol{c})$. Rotating the crank of the Peaucellier inversor, S traces out a straight line segment, and the locus of P is a part of $\mathcal{C}$.

It remains to consider the choice of the bar lengths, obviously corresponding to the radius of the disc D in the preconditions of the *Universality Theorem*. We demand P to be freely movable in D, which immediately imposes the conditions $\max(m,n)-\min(m,n)\leq\inf\{d(O,p):p\in D\}$ and $m+n\geq\sup\{d(O,p):p\in D\}$, where $d(\cdot,\cdot)$ denotes the Euclidean distance.

This is easily possible by choosing $m=n=\frac{1}{2}\sup\{d(O,p):p\in D\}<\infty$, since D is bounded. For a fixed choice of m and n, all intermediate links of the linkwork are restricted to be in a bounded region of the plane since P stays inside a closed disc of radius $m+n$ around O. Thus, we can select the size of the links as needed during the construction. This completes the proof of *Kempe's Universality Theorem*. □

For the sake of completeness, we remark that by iteratively enlarging m and n we can draw any region of an algebraic curve. This gives a simple reformulation of

Kempe's Universality Theorem (alternative version)**.** *Let $f\in\mathbb{R}[x,y]$ be a polynomial defining an algebraic curve $\mathcal{C}=\mathcal{V}(f)=\left\{(x,y)\in\mathbb{R}^2:f(x,y)=0\right\}$ in the plane.*

Then there exists a series of linkworks s.t. for any bounded region $\mathcal{S}\subset\mathbb{R}^2$ the series contains a linkage translating a motion of a point S along a straight line segment to a motion of a point P along $\mathcal{C}\cap\mathcal{S}$ and vice-versa. □

By applying Stone-Weierstraß approximation theorem, we can further conclude the

Sign Your Name Theorem (Thurston)**.** *There exists a series of linkworks signing your (finite) name in arbitrary precision.*

3.3 Complexity of the Kempe Linkage

Kempe himself states in the original publication of his work [Kem76]

> It is hardly necessary to add, that this method would not be practically useful on account of the complexity of the linkwork employed, a necessary consequence of the perfect generality of the demonstration. The method has, however, an interest, as showing that there *is* a way of drawing any given case ; and the variety of methods of expressing particular functions

> that have already been discovered renders it in the highest degree probable that in every case a simpler method can be found. There is still, therefore, a wide field open to the mathematical artist to discover the simplest linkworks that will describe particular curves.
>
> The extension of this demonstration to curves of double curvature and surfaces clearly involves no difficulty.

While Kempe was wrong with his last sentence, (see section 3.4) he probably is perfectly right with the first part.

Gao et al. presented [GZCG02] a proof of a complexity of $\mathcal{O}(d^4)$ bars for a linkage describing a curve of degree d, which turned out to be clearly overestimated. We will—according to the work of Abbott et al. [ABD08]—proof the number of bars to be in $\mathcal{O}(d^2)$, which is optimal.

First, we state that the number of terms in formulation (3.1) of $\tilde{f}$ does not exceed d^2: there are at most $d+1$ possible choices of s and $2d+1$ choices of t, which means that the number of $\boldsymbol{a}_{s,t}$ and $\boldsymbol{b}_{s,t}$ sums up to $\mathcal{O}(d^2)$. Furthermore, we note that each of the basic linkages presented in chapter 2 and used in the construction only consists of a finite number of elements.

In the construction, for step 0 and 1 we obviously only have to use a constant number of bars to establish the coordinate system. The construction of the angles $s\varphi$ and $t\theta$ is done once for each value of s and t, totalling to $\mathcal{O}(d-2+2d-1) = \mathcal{O}(d)$. Then we need to use a number of angle adders matching the non-canceling terms $\cos(s\varphi + t\theta)$ and $\cos(s\varphi + t\theta - \frac{\pi}{2})$ in (3.1), which is in $\mathcal{O}(d^2)$. In step 5 each of those angles is attached a bar to scale it's length; this likewise is in $\mathcal{O}(d^2)$.

Finally, we need to sum up all the terms of $\tilde{f}$ by translators, which corresponds to the hinges resulting from step 5 and thus likewise is in $\mathcal{O}(d^2)$. Gao et al. used a slightly different approach on attaching the translators, following their further simplification of (3.1), and apparently thus miscounted the links needed.

Our results match those of Abbott et al. [ABD08], who prove a number of $\mathcal{O}\left(\binom{n+2m}{2m}\right)$ bars to be both sufficient and optimal for the tracing of the vanishing locus of a polynomial of total degree n in $\mathbb{R}[x_1, y_1, \ldots, x_m, y_m]$.

Note that this only refers to the mechanical complexity of the construction. For a computational solution we have to tackle two problems:

1. The linkwork essentially serves as an analog calculator; when we want to join two bars, we can just grab their endpoints and move them to a single valid point for join. In consequence, all other links of the mechanism not pinned to the

plane may change their position to fit the two boundary constraints. In the same manner, once the construction is done, a force on every element instantly alters the whole linkwork configuration accordingly.

In a simulation on a digital computer, this means that we have to solve a system of equations for all links simultaneously. In particular, finding an allowed configuration of bars satisfying that S lies on the line $\mathcal{V}(x + \boldsymbol{c})$ amounts to finding a zero of the polynomial f, which is intractable for higher degrees.

2. Even if we restrict ourselves to only move P and adjust the mechanism to fit, we have to take the complexity of the coefficients $\boldsymbol{a}_{s,t}$, $\boldsymbol{b}_{s,t}$ and $\boldsymbol{c}$ into account to just determine the bar lengths. Usually, we talk about a polynomial $f \in \mathbb{Q}[x, y]$ with rational coefficients, or, equivalently by multiplying with the largest common multiple of the divisors of the coefficients, $f \in \mathbb{Z}[x, y]$.

   In this case the necessary equations have to be solved either approximately, or the computation time has to be calculated depending of the coefficient complexity of the input, with respect to computing rationals in arbitrary precision or even irrational numbers, represented for example by isolating intervals and Sturm sequences.

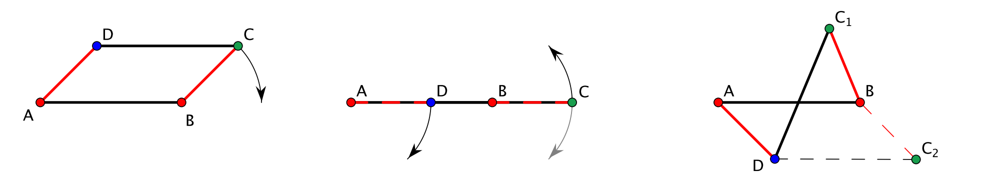
3.4 Generalizations and Criticism of Kempe‘s Proof and Open Questions

Abbott states in his thesis [ABD08], that

> Kempe [...] published a surprising proof that one could build a linkage such that a pen placed at a single vertex could draw the intersection of any algebraic curve with any closed disk. [...]
>
> Kempe‘s proof was flawed, however, because his constructions had additional configurations beyond those he intended them to have.

Why these apprehensions do not impact our work will be explained in the next chapter; here it shall suffice to give an example of the most simple linkage failing, and the idea on how to solve this.

Consider a parallelogram linkage $ABCD$ as shown in figure 3.3, which is used all over the place in translators. If now the linkwork is elongated to it‘s maximum extent, both Kempe and we expect the motion to continue by a rotation of C and D around



(a) parallelogram (b) degenerate position (c) contraparallelogram

Figure 3.3: Three possible configurations of a simple translator linkage

B or A *in the same direction*. However, from the theoretical point of view there is no reasoning for this assumption, although in reality inertia or other arguments of physics may make it appear as such.

Therefore, Abbott et al. propose to use a braced parallelogram, which consists of an additional bar, connecting the midpoints of two opposite sides (figure 3.4). This simple

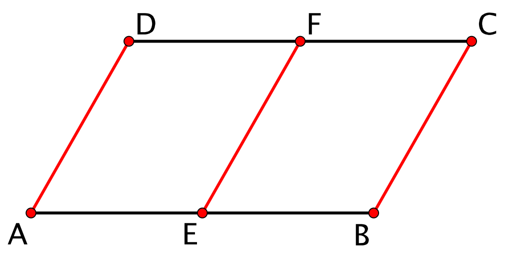


Figure 3.4: A braced parallelogram

addition, which has no influence on the order of the complexity, effectively avoids the problem:

In a (nondegenerate) contraparallelogram configuration, $EF = \frac{AC+BD}{2}$, where E and F denote the midpoints of the sides AB and CD. Now assume there is an intersection X of AD and BC. By the triangle inequality,

$$\begin{aligned}2\,|EF| = |AC| + |BD| &< (|AX| + |XC|) + (|BX| + |XD|)\\ &= (|AX| + |XD|) + (|BX| + |XC|)\\ &= |AD| + |BC|\\ &= 2\,BC,\end{aligned}$$

but by definition of E and F we know $|EF| = |BC|$; therefore, only the degenerate antiparallelogram configuration exists, which at the same time is a degenerate parallelogram and thus allowed. In a similar manner, Abbott et al. extend the reversor linkage to stay a contraparallelogram.

Furthermore, they discuss the concept of "rigid continuous constructability", which probably matches the intuition of Kempe's original work. The term refers to mechanisms that delineate a given connected set $\mathcal{S}$ *continuously*, i.e.—in analogy to the term in analysis—without sudden reconfigurations of the linkage during the drawing of the set, and *rigidly*, which means that for every point $p \in \mathcal{S}$, there are only finitely many configurations of the linkage s.t. p is represented by the linkage. In consequence, the combination of both properties means that a linkage has no motions other than those absolutely necessary to delineate $\mathcal{S}$. For a further examination of the topic, we refer to [ABD08].

While rigid constructability is not an issue in the Euclidean plane, it turned out to be a tough one for higher dimensions. Neither Abbott et al. nor King [Kin98], who analyzed the topic using a different kind of linkages, have so far been able to solve the question which drawable sets in dimension $n > 2$ are rigidly constructible, although they independently proved algebraic varieties, in particular algebraic curves in n-space and hypersurfaces, to be continuously constructible by series of linkworks.

Finally, it is not yet known whether there are linkages delineating an arbitrary large part of a connected component of general curves in a single smooth motion of the crank of the Peaucellier cell constraining S to $\mathcal{V}(x + \boldsymbol{c})$. Intuitively this means that, while all points on $\mathcal{C}$ can be reached during or at the ends of continuous motions, there can be branches in the configuration space s.t. we have to "turn around" and redo a motion on $\mathcal{V}(x + bc)$ to get a different outcome of the arrangement of bars.

4 Simulation of Kempe Linkages using Cinderella and Xalci

The major part of this thesis is the developement of a simulation of Kempe's line-curve-translation, featuring the dynamic geometry system *Cinderella* and the algebraic curve visualization of *Xalci*. The results of this work discussed throughout this chapter can be found online at `http://www.math.uni-sb.de/ag/schreyer/Kempe-Linkage/`.

As mentioned in section 3.3, linkworks exceed the classical calculation model by simultaneously performing movements according to the constraints induced by the lengths of the rigid bars, corresponding to equation solving in computing, in essentially no time. While this is theoretically possible to mimic on computers, current calculating capacities allow to do so for none but very easy examples in real-time due to the large number and complexity of the constraints involved. Arbitrary deformation models are out of reach even for state-of-the-art techniques.

Consequently, all existing DGS including *Cinderella* use an acyclic dependency graph approach, where every elements position only influences the construction parts contained in it's childrens layers. Accordingly, it is not possible to move elements whose position is fixed without any degree of freedom by boundary constraints, as is with the point S on the straight line resulting from Kempe's construction.

On the other hand, the applications allow to easily state preconditions which aren't obvious in mechanism design. For example, restricting a point to be free only along a straight line is the most simple thing in the world of dynamic geometry applications.

Accordingly, we do not try to implement the linkages in every detail, but instead developed a modified version adapted to fit the needs of a dynamic construction simulation.

4.1 Some Design Aspects of Cinderella

Throughout this setion, we will discuss some key features of *Cinderella* which influence our implementation. The statements mainly rely on Ulrich Kortenkamp's doctoral thesis "Foundations of Dynamic Geometry" [Kor99] from 1999, in which he presents the theory behind *Cinderella*. The scripting abilities have been added later, thus their covering is based on the on-line documentation of the program [KRG].

Most of the mentioned characteristics which distinguish *Cinderella* from other approaches, are controversely discussed, especially among the educational community, providing plenty of information material to any interested reader.

4.1.1 Complex Projective Geometry

To achieve the most comprehensive insight in geometric theorems possible, all coordinates and intermediate values in *Cinderella*'s computations are evaluated in the complex projective plane. Although the viewport naturally only covers a part of the real

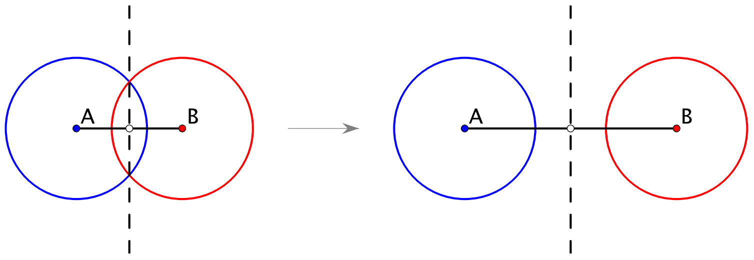
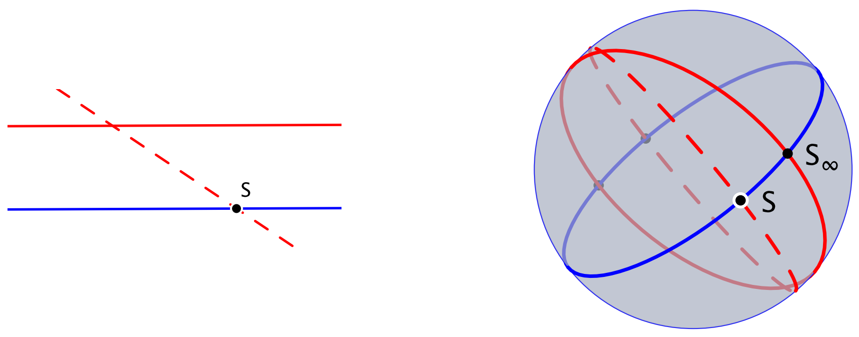


Figure 4.1: The perpendicular bisector of a segment, constructed as the line through the intersection points of two circles

Euclidean plane, this is a highly useful feature to have, despite it's irritating appearance at the first glance. For example (see figure 4.1), the perpendicular bisector of a segment can be constructed by drawing a line through the intersections of two equally sized circles around the endpoints. If now during a construction process the endpoints alter their position and the length of the segment exceeds the diameter of the circles, the intersections leave the real plane. However, the complex affine linear subspace defined by their difference vector still includes the real bisector and is thus displayed as such.

A related topic is the consideration of elements at infinity in the projective space. A simple example is the intersection of two lines, as depicted in figure 4.2. When the dashed line rotates s.t. it becomes parallel to another line, their intersection point is still well-defined in the projective plane. *Cinderella* not only handles this case internally, but is also able to show the situation in a spherical viewport.

Together those approaches allow a very thorough handling of constructions. Many degenerate cases in the real affine plane turn out to be perfectly straightforward in complex projective geometry. In particular, the user may specify constructions which



(a) Euclidean view (b) spherical view

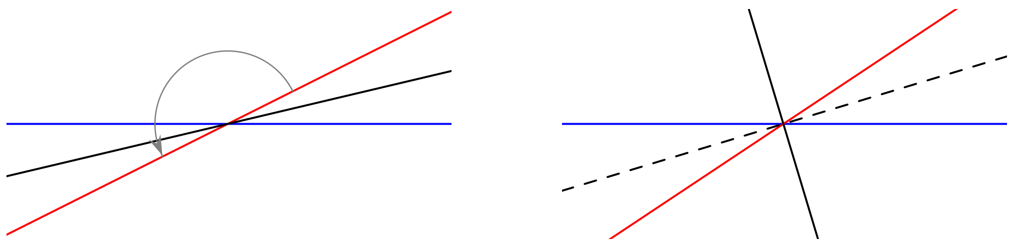
Figure 4.2: The intersection of two parallel lines at infinity

are considered to be invalid by other applications.

Automatical input generation benefits in additional extent, because a design can be done in one go and moved to a nonsingular position later. The treatment of degenerate cases in $\mathbb{R}^2$ is left to *Cinderella* and cannot cause any inconsistencies.

4.1.2 Continuity

As the term "dynamic geometry" suggests, constructions can be modified once they are established. The users then expect a construction to behave "nicely" when input elements are moved, which means that no sudden jumps occur when the input is changed continuously. The situation in figure 4.3 serves as an example for the behaviour of a *continuous* geometry system versus so called *deterministic* approaches. The black line is defined as the angular bisector of the blue and red line, a geometric primitive offered by any system.



(a) starting position (b) situation after a rotation of the red line by 180 degrees

Figure 4.3: The angular bisector in continuous and deterministic geometry systems

When the input changes s.t. the red line performs a rotation by 180 degrees, *Cinderella* continuously rotates the solid black line with half the speed. In constrast, many other programs define the angular bisector as the line dividing the angle of smallest measure, leading to an abrupt change of place when the red line passes the fixed blue, and resulting in the dashed line. Obviously, such incontinuities are a highly undesired performance in linkage simulation.

While some other implementations are—to variable extent—upgraded to realize continuous motions, *Cinderella* uses a technique called "complex tracing" to not allow discrete jumps in the first place. On the other hand, this implicates that it is not possible to specify, for example, the angular bisector of the smallest angle, even if this is what the user really wants.

Although this feature is not directly used in our application, the continuity also means that *Cinderella*'s locus tracing algorithm usually is able to draw the complete loci even of complicated traces, while other programs are restricted to the curve parts reachable within the tighter constraints of their object definitions. This is even intensified by the inclusion of complex and infinite elements discussed in the previous subsection.

4.1.3 Automated Theorem Checking

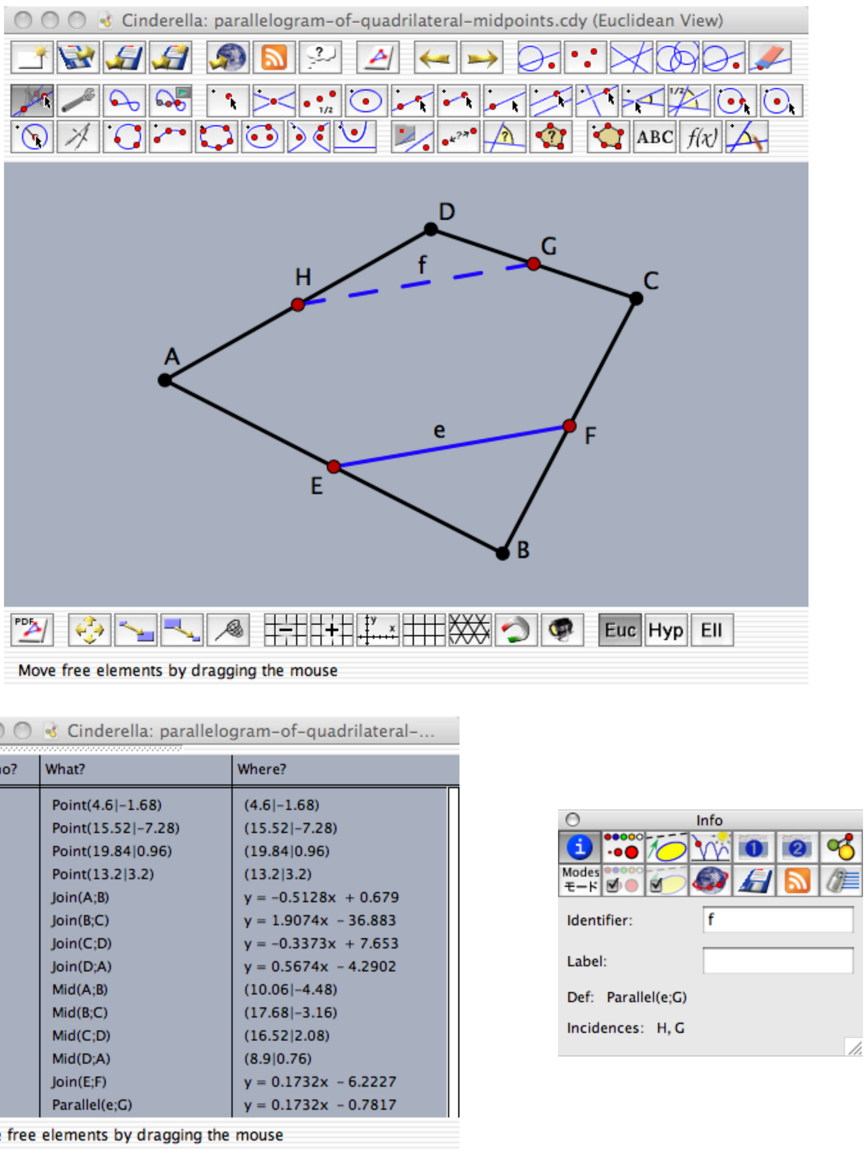
A further unique selling point of *Cinderella* is an integrated theorem checker.

For every construction you can think of there are plenty of ways to describe the desired result—*Cinderella* is able to decide whether they are equivalent.

As an example, assume we are given an arbitrary quadrilateral $ABCD$ and the midpoints E, F, G and H of it's sides. There is a theorem stating that then $EFGH$ is a parallelogram. This fact is recognized by *Cinderella*, as shown in the screenshots 4.4. In the left corner, below the Euclidean viewport, we see *Cinderella*'s construction text window, which, besides the current position of the elements, shows their definition (the "What?" column). Note that the order of elements corresponds to their construction; each object is defined only by preceding objects. In particular, this means that definitions have to be acyclic; no operation allowed in *Cinderella* other than dragging inputs with some degree of freedom can change the currently existing design.

Finally, in the lower right the information about a single element is shown. Here the line f is selected, along with it's definition as the parallel to e through G. *Cinderella* automatically detects that f is incident to H, in addition to the trivial incidence with G.

The gathering of this information does not rely on a given set of theorems. Instead, *Cinderella* randomly generates several possible instances continuously reachable from the construction with respect to the definitions of elements. If an incidences is consistent


Cinderella: parallelogram-of-quadrilateral-midpoints.cdy (Euclidean View)
D
G
H
f
C
A
e
F
E
B
PDF
Euc
Hyp
Ell
Cinderella: parallelogram-of-quadrilateral-...
Info
Modes
モード

Move free elements by dragging the mouse

	Who?	What?	Where?
●	A	Point(4.6|-1.68)	(4.6|-1.68)
●	B	Point(15.52|-7.28)	(15.52|-7.28)
●	C	Point(19.84|0.96)	(19.84|0.96)
●	D	Point(13.2|3.2)	(13.2|3.2)
/	a	Join(A;B)	y = -0.5128x + 0.679
/	b	Join(B;C)	y = 1.9074x - 36.883
/	c	Join(C;D)	y = -0.3373x + 7.653
/	d	Join(D;A)	y = 0.5674x - 4.2902
🔴	E	Mid(A;B)	(10.06|-4.48)
🔴	F	Mid(B;C)	(17.68|-3.16)
🔴	G	Mid(C;D)	(16.52|2.08)
🔴	H	Mid(D;A)	(8.9|0.76)
/	e	Join(E;F)	y = 0.1732x - 6.2227
/	f	Parallel(e;G)	y = 0.1732x - 0.7817

Move free elements by dragging the mouse

Identifier: f

Label:

Def: Parallel(e;G)

Incidences: H, G

Figure 4.4: Automatic theorem checking in *Cinderella*

over all these tries, it is reported as proven. The results of the checks are used to some extent to avoid unnecessary duplications of constructions and, thus, saving computation time.

The number of tests performed is chosen to match an undocumented, but seemingly very high probability. Since all calculations in *Cinderella* are done with floating point arithmetics, Kortenkamp however has to admit that the theorem checker can be fooled by constructions involving extreme intermediate values.

The combination of continuity guarantees and intrinsic theorem checking allows to take up Abbott et al.'s criticism on Kempe's elementary linkages explained in section 3.4 from another point of view. The addressed problems of the occurences of additional configurations are automatically avoided by *Cinderella*, since it's continuity model simultaneously applies to all elements of the construction.

As long as we build upon a nondegenerate state in the beginning, incidences such as the parallelity of the bars of the translators remain invariant under arbitrary movements. Thus, even if we don't define the parallelogram linkage by the geometric primitive offered by the system, but position the hinges, say C of figure 3.3, on one of the two intersection points of circles determining the bar lengths, *Cinderella* will remember this choice throughout the construction.

4.1.4 CindyScript

An implementation of automated linkage generation would not have been possible without some interface to instruct *Cinderella* which operations to perform. This is able by the use of *CindyScript*, a functional high level language interpreted by *Cinderella*. Besides common features of any object-oriented programming language it allows to interactively modify the construction data. For this purpose both a command shell and a script editor are offered; the latter can react on events like user input or redrawing of the viewport.

The CindyScript interpreter is also part of *Cinderella* applets, thus predefined scripts can also be used in web presentations. In addition, the authors recently added an interface allowing foreign applets to execute CindyScript commands from the outside. Our implementation almost exclusively uses this way of communication with *Cinderella*.

This feature still is in the testing stage, and it turned out to have several issues not yet completely eliminated. In particular, *Cinderella* currently is not able to reliably give feedback to the outside after execution of an instruction. As a necessary consequence, our implementation controls *Cinderella* through a blindfold. Since we intendedly do not try to double the geometric computations and cannot inspect the internal information of *Cinderella*, we are thus not yet completely able to adapt input parameters such as the dimensions of the linkwork according to the restrictions of the viewport or the given curve.

4.2 Visualization of Implicit Algebraic Curves using Xalci

The nature of *Cinderella* does not allow the dynamic realization of a Kempe mechanism in full extent: we are able to see the linkage working when we move our free point P, but the hinge S is uniquely determined by the rigid bars. Even worse: the linkage usually is consistent even if P does not lie on the curve. Thus the least we need for a proper simulation is an image of the curve to get a clue which path we have to move P on.

Xalci provides this, and more. The Algorithms and Complexity working group at the Max-Planck-Institute for Computer Science in Saarbrücken developed a tool to analyze the topology of an arrangement of implicit algebraic curves in $\mathbb{Q}[x, y]$. The implementation features detection of singularities and extrema of curves and provides the position of intersections of several curves. All necessary calculations are guaranteed to give topological correct results, and the coordinates theoretically can be requested in arbitrary precision.

While we have no use for curve arrangements since our implementation of Kempe's linkwork only deals with one polynomial, the rasterization power of *Xalci* offers great opportunities for our simulation. Utilizing the topological analysis of a curve, *Xalci* can decide which regions the arcs of the curve lie in. Starting from the equally known feature points of the curve, those arcs can then be traced out and rasterized.

The time-consuming part here is the dissection of the curve, which only has to be done once. Afterwards, the rasterization runs in linear time in terms of both the number of arcs and the horizontal resolution of the output. This high efficiency convinced the researchers to provide a Flash-based web interface, to be found at `http://exacus.mpi-inf.mpg.de/cgi-bin/xalci.cgi`, as an appetizer for the work to come.

Since there is no stand-alone version of *Xalci* publicly available yet, Pavel Emeliyanenko kindly agreed to support our wish to use the web interface as a backend for visualization purposes, and tailored an output option to our particular needs. This not only allows us to to draw the curves, but by reattaching the arc segments we can gather path information used for automatic smooth animation of the movement of P along the curve. *Xalci* actually seems underchallenged by far by this task—while usually the correct visualization of implicit curves is considered a very expensive computation, neither *Xalci* nor the web transport turned out to be a bottleneck in our simulation.

The interface still might be a subject to change during the further developement of *Xalci*.

4.3 Geometric Primitives in Linkage Simulation

As mentioned in the introductory paragraphs to this chapter, in a DGS like *Cinderella* you do not have the possibilities nor the needs to mimic a linkwork in every detail. However, we can simulate all geometric computations performed during the construction algorithm presented in section 3.2.

Since arbitrary reals cannot be represented in *Cinderella*, we demand the input polynomial to have rational coefficients, as well as $m, n \in \mathbb{Q}$. A quick look on the transformations done to obtain $\tilde{f}_{m,n}$ from f shows that all coefficents $\boldsymbol{a}_{s,t}$, $\boldsymbol{b}_{s,t}$ and $\boldsymbol{c}$ are rational, too. The reader might ask whether this is really necessary because *Cinderella* uses floating-point arithmetics, so results will be inexact anyway, and he will have a point there. Approximating arbitrary reals will yield the same precision as our approach. But our goal is a theoretically correct implementation, which at least has to feature correct algorithms, even if they cannot be handled in full extent by the tools used, and there are lengths of irrational measure (such as π) not constructible with the primitive operations allowed by any ruler and compass approach regardless of the underlying implementation.

Our implementation proceeds in the same steps as Kempe's description:

0. - We start by establishing the coordinate system by defining points $O = (0, 0)$ and $X = (1, 0)$ at fixed coordinates and draw the unit circle through X around O.

     P is initialized to an arbitrary starting position within the open set reachable by the bars avoiding the degenerate case where the supporting lines of the bars overlap. Obviously, a preferable choice is some point on the curve; since P is free to move anyway, we can safely use the positioning facilities offered by *Cinderella*.
   - Since m and n are rational numbers, they are constructible, meaning there are ruler-compass constructions to find a point mX with distance m to O and $m - 1$ to X, only depending on O and X.[1] The procedure allowing

[1]A more formal definition of the constructability of x is that x is contained in an iterated algebraic field

to generate mX and nX is presented in step 5 and does not depend on intermediate elements; so we can for now assume to have given points from which we can take the lengths m and n to the origin and draw circles of radius m around O and n around P.

One of their intersection points is chosen as a possible instance of A_1. Joins, i.e. defining a line (achieved by Peaucellier cells in linkages), are primitive operations in a DGS as well as the drawing of parallels through given points, so we can determine the point B_1 as the intersection of the parallels to OA_1 through P and A_1P through O.

1. In *Cinderella*, contrary to deterministic geometry systems, we have to face the same problems regarding the orientation of the y-axis discussed in section 3.2. Just choosing a third free point will prevent the program to apply it's incidence checking on elements depending on the axis, since the defining third point might move later. Thus we have to take any of the two intersections of the unit circle and the y-axis (defined as the perpendicular on OX through O) for Y.

   This is just mentioned for the sake of completeness and does not cause any trouble, because after all the same instruction sequence will always cause the same result to appear on the screen.

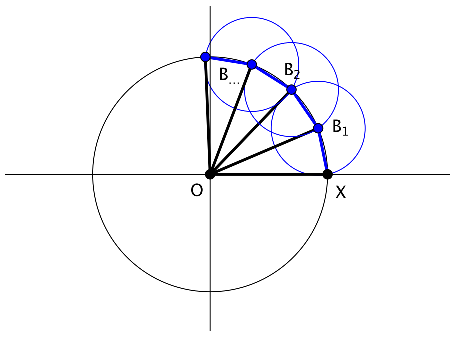
2. We now generate points on the unit circle representing the angles φ and θ. This can be done by dividing the distances of A_1 and B_1 to O by m and n; again, we refer to step 5.

   For clearness, in figure 4.5 (a) the points are already assumed to be on the unit circle. Then to yield a multiple of an angle (θ, corresponding to B_1, in the figure) it is sufficient to iteratively draw circles around B_t through B_{t-1}, starting with B_1 and X. There is at most one additional intersection of these circles with the unit circle besides B_{t-1}; this point gives B_{t+1} with angle $\measuredangle XOB_{t+1} = (t+1)\theta$ by congruence of $\triangle B_{t-1}OB_t$ and $\triangle B_tOB_{t+1}$.

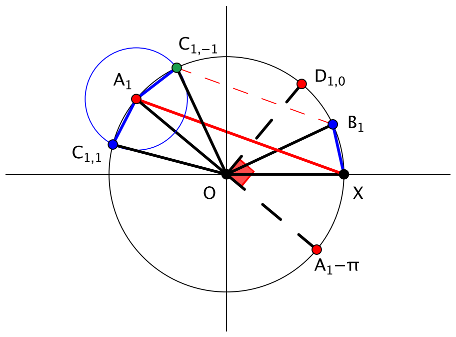
   If there is no additional intersection, $\measuredangle B_{t-1}OB_t = \pi$, which is a degenerate case not induced by boundary constraints other than the position of P, which is

extension of degree two of $\mathbb{Q}$. The field operations $+$, $-$, $\cdot$ and $\div$ are linear operations, which can be achieved by the ruler; additionaly, the constructible numbers are closed under $\sqrt{\cdot}$, which corresponds to the solving of polynomial equations in degree two, or, geometrically, intersections of circles and lines or circles.

All operations but root extraction involved are used and presented throughout this construction; for a more concise treatment of the topic see e.g. [Lab08] or introductory textbooks on algebraic number theory.



(a) multiplication of angles



(b) addition of angles

Figure 4.5: Multiplication and addition of angles

allowed to change. *Cinderella*'s theorem checker thus finds valid instances where B_{t+1} is well-defined, and allows the construction of "the intersection other than B_{t-1}" nevertheless.

3. Addition of angles (figure 4.5 (b)) follows a similar idea. To construct a $C_{1,1}$ as a representant of $\varphi + \theta$, we take the distance XB_1 by the compass and draw a circle around A_1.

   Now there really are two intersections between circles not yet known, so we need additional information to decide which one corresponds to addition and which to subtraction. We thus take the second intersection but B_1 of the unit circle and a parallel to XA_1 through B_1 and anticipate in calling it $C_{1,-1}$, the subtraction result. Now we know $|XB_1| = |A_1C_{1,-1}|$, so $\triangle XOB_1$ and $\triangle C_{1,-1}OA_1$ are congruent, and we conclude $\measuredangle XOC_{1,-1} = \measuredangle XOA_1 - \measuredangle C_{1,-1}OA_1 = \measuredangle XOA_1 - \measuredangle XOB_1 = \varphi - \theta$, thus $\measuredangle XOC_{1,1} = \measuredangle XOC_{1,-1} + \measuredangle C_{1,-1}OC_{1,1} = \varphi - \theta + 2\theta = \varphi + \theta$ for the remaining intersection $C_{1,1}$.

4. Since we are given an representant Y for $\frac{\pi}{2}$, subtraction of $\frac{\pi}{2}$ can be achieved in the same manner, starting from the mirror point of A_1 w.r.t. O and adding $\frac{\pi}{2}$.

5. Now we consider the multiplication of lengths by rationals, say the construction of $\frac{a}{b}X$ from X, where $a, b \in \mathbb{Z}$. The constructions can be rotated arbitrarily and generalize for the scaling of the $C_{s,t}$ and $D_{s,t}$ by $\boldsymbol{a}_{s,t}$ and $\boldsymbol{b}_{s,t}$.

   We start with integral multiplication of X to aX and bX, to later divide two lengths. A simple, but ineffective approach on integral multiplication is iteratively adding OX using circle constructions. This is sufficient for the angle multiplications, where typically the highest multiple (corresponding to the degree d of

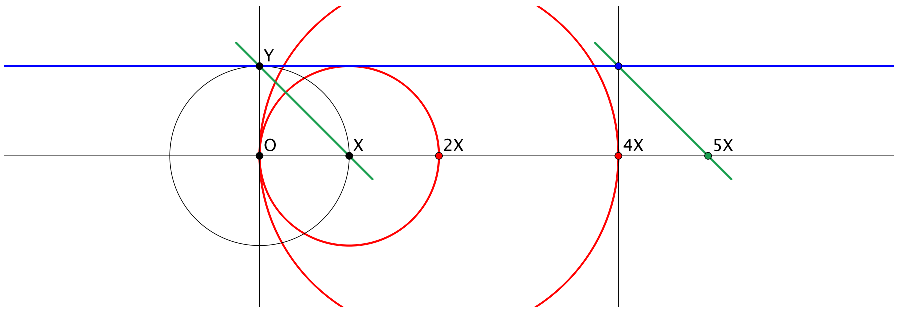


Figure 4.6: Multiplication of lengths

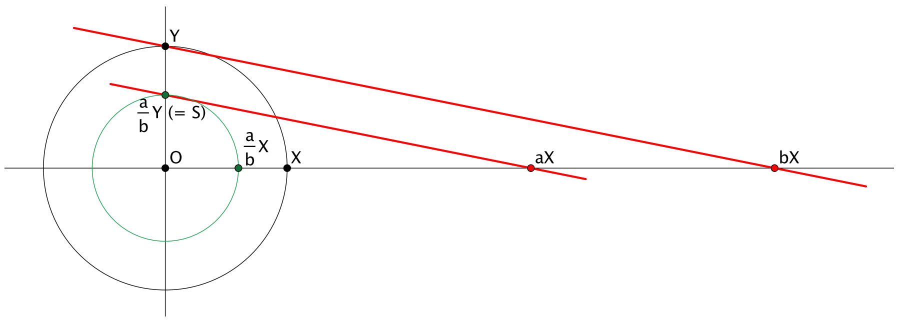


Figure 4.7: Division of lengths by application of the intercept theorem

the polynomial f) is rather low; however, this usually is not the case for the coefficients.

Instead, we first construct all powers of two $2^i X$ using circles in corresponding sizes as shown in figure 4.6. Then we add the required powers to get the final result ($5X$ here). This is done using the depicted parallels construction, equivalent to the translator linkage, which moves OX (or some arbitrary multiple) to $4X$, yielding $5X$. Thus we achieve a reduction of the construction complexity from $\mathcal{O}(a)$ to $\mathcal{O}(\log a)$, compared to the trivial solution.

Division is allowed by the intercept theorem, as depicted in figure 4.7. Starting from aX and bX, we join the divisor bX with Y. The intersection S of the parallel though the dividend aX with the y-axis satisfies $\frac{a}{b} = \frac{|OaX|}{|ObX|} = \frac{|OS|}{|OY|}$, thus $S = \frac{a}{b}Y$. Rotation back to the x-axis yields $\frac{a}{b}X$.

Note that the orientation of the y-axis matters for none of the constructions mentioned above, so we are free to use $-Y$ instead if we are not yet able to decide which point corresponds to the angle rotated by 90 degrees *clockwise*. For a length not coinciding with the x-axis, we imagine any of the two intersections of a perpendicular through O and the unit circle as a local Y. This is especially important for getting the position of the points A_1 and B_1 on the unit circle in step 1.

6. Finally, the easiest step throughout the simulation is the summation of all occuring terms. Since we do not need to care about fixed bar lengths—those are automatically adapted by the DGS—we can employ the simple parallelogram construction of figure 2.4 (a).

4.4 Presentation of Our Simulation of Kempe Linkages

We want to complete this chapter with a presentation of our implementation, accessible to anyone interested at `http://www.math.uni-sb.de/ag/schreyer/Kempe-Linkage/`.

Our solution is written in Java and designed as an applet, a decision made to ease the communication with *Cinderella*. The exact rational arithmetics needed for the transformation of f to $\tilde{f}_{m,n}$ are provided by the free open source JScience library [JSc]. The complete application amounts to approximately 6500 lines of code, including the *Xalci* interface; sources are available on request.

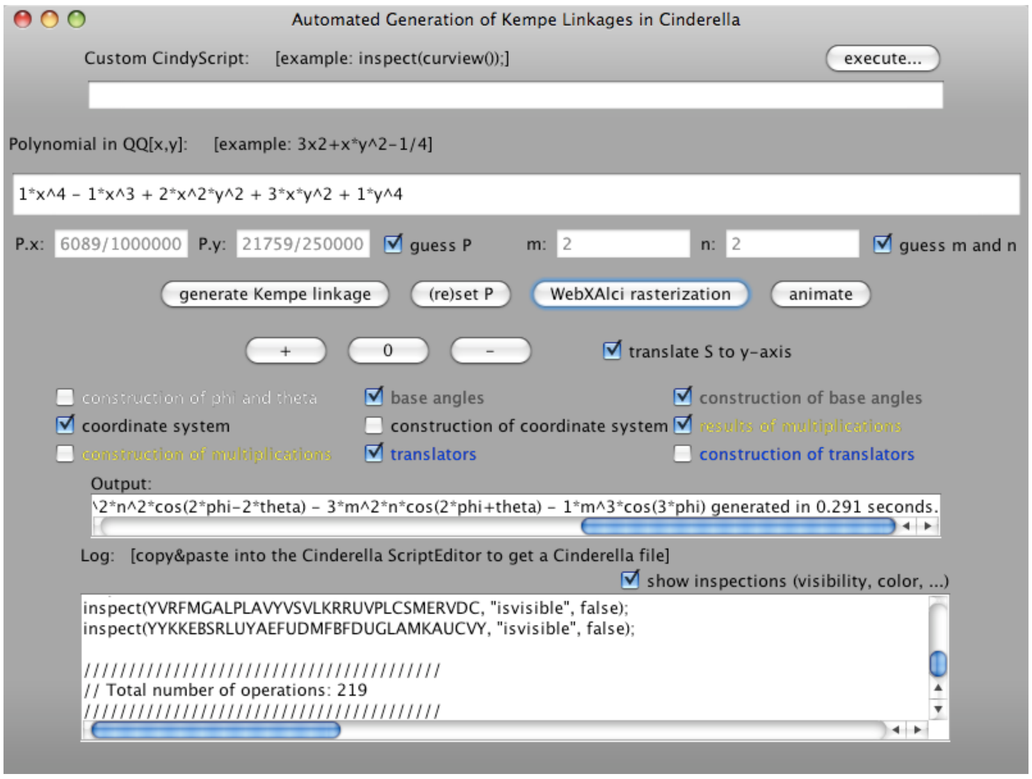
4.4.1 The User Interface

Figure 4.8 shows the user interface of our program, giving an overview of the functionality.

Using *Xalci*'s rasterization of the curve, we are able to approximate coordinates of a random point P on the curve within the current viewport. Consequently, we offer automatical adaption of the bar length s.t. P is reached without forcing the initial parallelogram to an excessive elongation.

Up to now, there is no support for shifting the viewport of a *Cinderella* applet from outside, nor can we obtain the coordinates of the shown area. As a necessary consequence, we made the obvious decision to fix the view around the origin of the coordinate system. Therefore we offer the translation of the straight line S moves on to the y-axis, a part of which is always visible. For a minimum of convenience, we are able to allow arbitrary zooming.

Due to the complexity of the construction the output tends to become extremely messy; to face this inevitable problem, several intermediate construction steps can be hidden from the user. The defaults are reasonably chosen to get an idea of the dynamics of the construction without to much clutter (figure 4.9).



Custom CindyScript: [example: inspect(curview());]

Polynomial in QQ[x,y]: [example: 3x2+x*y^2-1/4]

```
1*x^4 - 1*x^3 + 2*x^2*y^2 + 3*x*y^2 + 1*y^4
```

P.x: 6089/1000000 P.y: 21759/250000 [✓ guess P] m: 2 n: 2 [✓ guess m and n]

[✓ translate S to y-axis]

construction of phi and theta [✓ base angles] [✓ construction of base angles]
[✓ coordinate system] construction of coordinate system [✓ results of multiplications]
construction of multiplications [✓ translators] construction of translators

Output:

```
\2*n^2*cos(2*phi-2*theta) - 3*m^2*n*cos(2*phi+theta) - 1*m^3*cos(3*phi) generated in 0.291 seconds.
```

Log: [copy&paste into the Cinderella ScriptEditor to get a Cinderella file]

[✓ show inspections (visibility, color, ...)]

```
inspect(YVRFMGALPLAVYVSVLKRRUVPLCSMERVDC, "isvisible", false);
inspect(YYKKEBSRLUYAEFUDMFBFDUGLAMKAUCVY, "isvisible", false);

/////////////////////////////////////////
// Total number of operations: 219
/////////////////////////////////////////
```

Figure 4.8: The graphical user interface of our implementation

For a deeper insight to the construction, we log all steps done throughout the construction in the CindyScript instruction format. Thus the user can redo the operations in a stand-alone instance of *Cinderella* and examine the simulation in every detail, although without our interface to *Xalci*. For short requests, the user can also use a CindyScript command line inside our application.

4.4.2 Animations

Sadly, a printout cannot express the dynamic nature of an animation very well. At least we can give some screenshots throughout a simulation (figures 4.10 and 4.11). We trace the motion of S while P moves along the curve $\mathcal{C} = \mathcal{V}(y^2 + x^3 - x^2)$; our setup uses $m = n = 1$. In the trigonometric form of the polynomial, no offset by constant terms occurs, so the line S stays on is the y-axis.

When P enters the viewport into the second quadrant, S comes along from above. Through the self-intersection of the curve in the origin, S continuously passes $(0, 1)$, to oscillate around the origin while P describes the loop of $\mathcal{C}$. When P leaves the

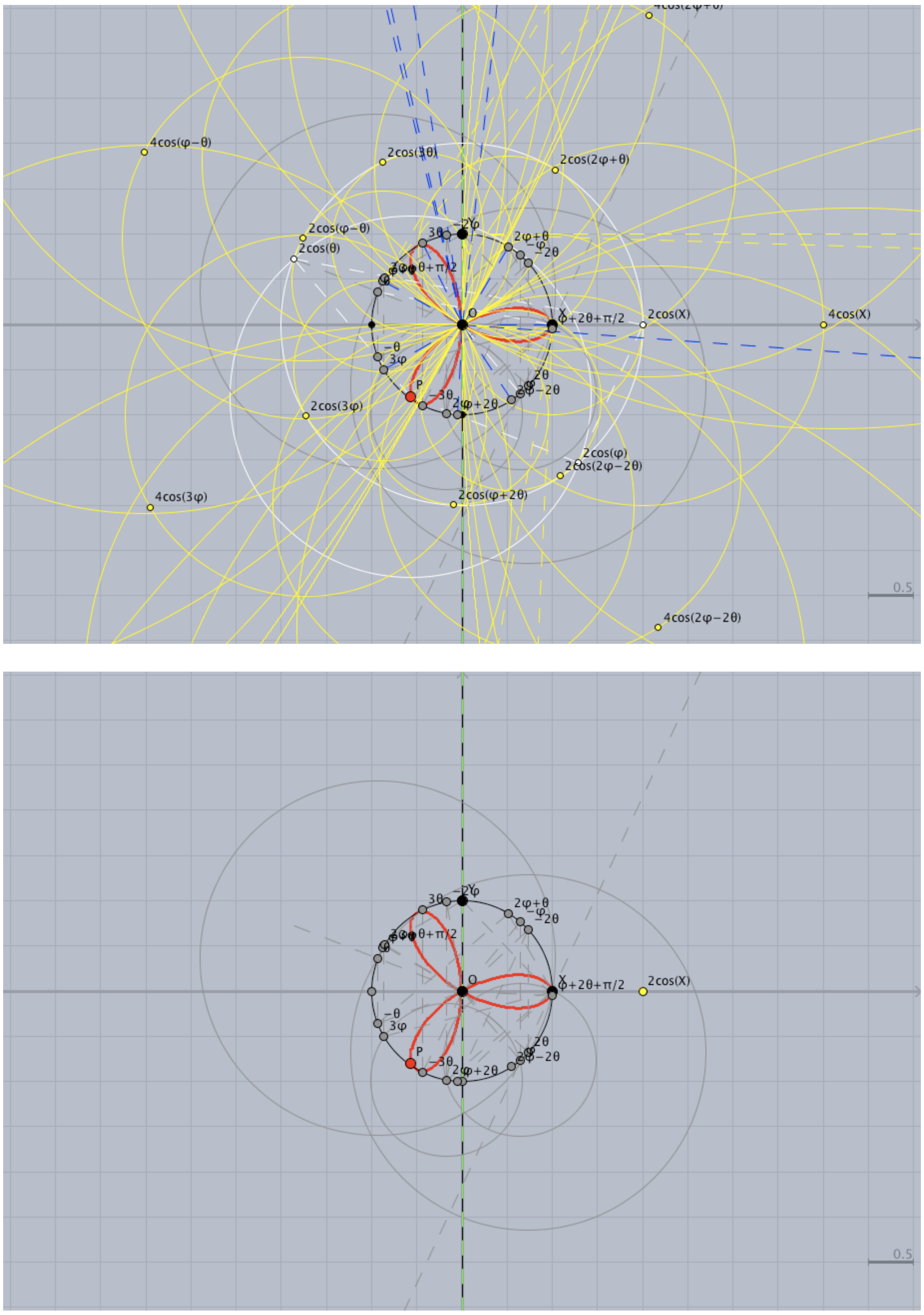


Figure 4.9: A view of all intermediate elements of a construction, in contrast to a more tidy version.

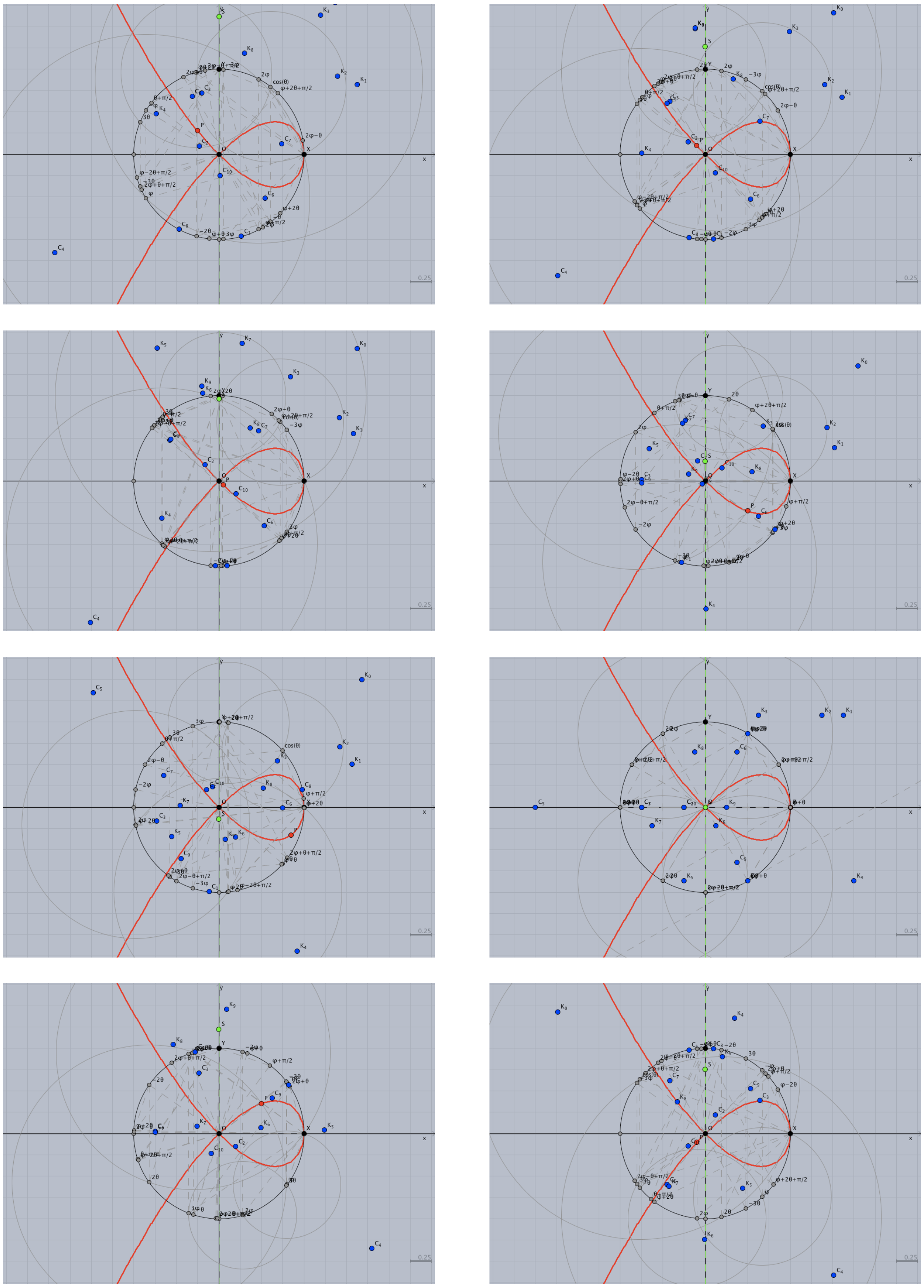


Figure 4.10: Animation stills of a Kempe linkage simulation of $y^2 = x^2 - x^3$

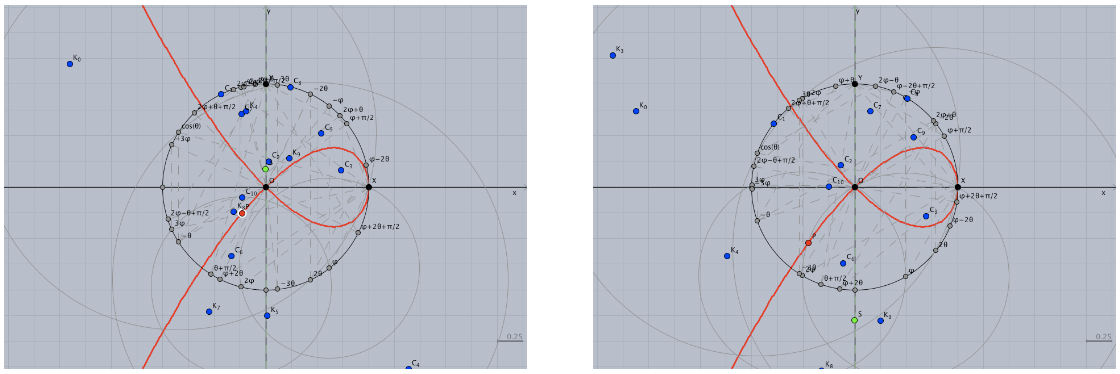


Figure 4.11: Animation stills of a Kempe linkage simulation of $y^2 = x^2 - x^3$ (cont'd)

viewport along the path in the third quadrant, S moves down towards $(0, -\infty)$. Since a construction mirrored w.r.t. the x-axis is equivalent, for another choice of a starting coordinates of P we may get the reflected linkage.

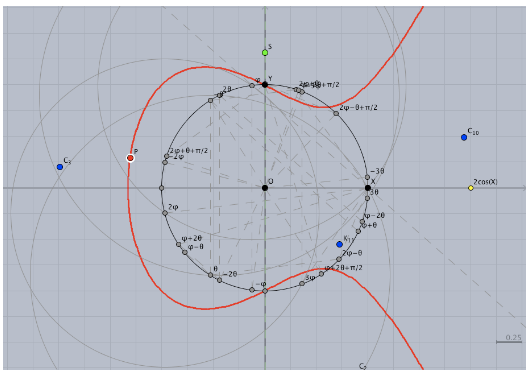
4.4.3 Complicated Curves and Statistics

Our implementation allows the input of arbitrary polynomials in $\mathbb{Q}[x, y]$, although the constructions may not be recognizable anymore.

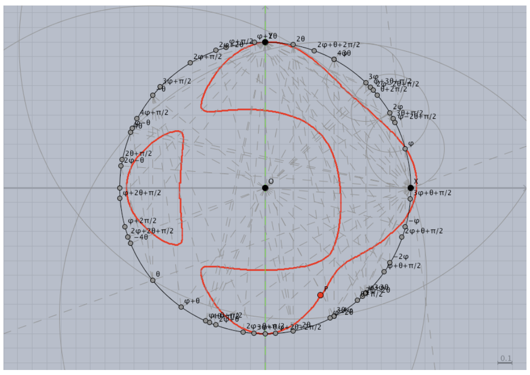
Inspection in stand-alone *Cinderella* proved the correct handling of singularities like the isolated origin point on the curve $y^2 = x^3 - x^2$. For every direction in which P moves out of the origin S leaves it's straight line; the configuration describing P is only continuously reachable along paths in the complex plane when the real part of the S' x-coordinate shall remain on the line.

Curves which cannot be rationally parametrized like the smooth elliptic curve in figure 4.12 (a) impose no difficulty on our approach. The instruction sequences for higher degree curves are generated well and have been checked for consistency for random examples up to degree eight; however, the applet design currently hardly allows reasonably smooth movements of complicated curves of degree four.

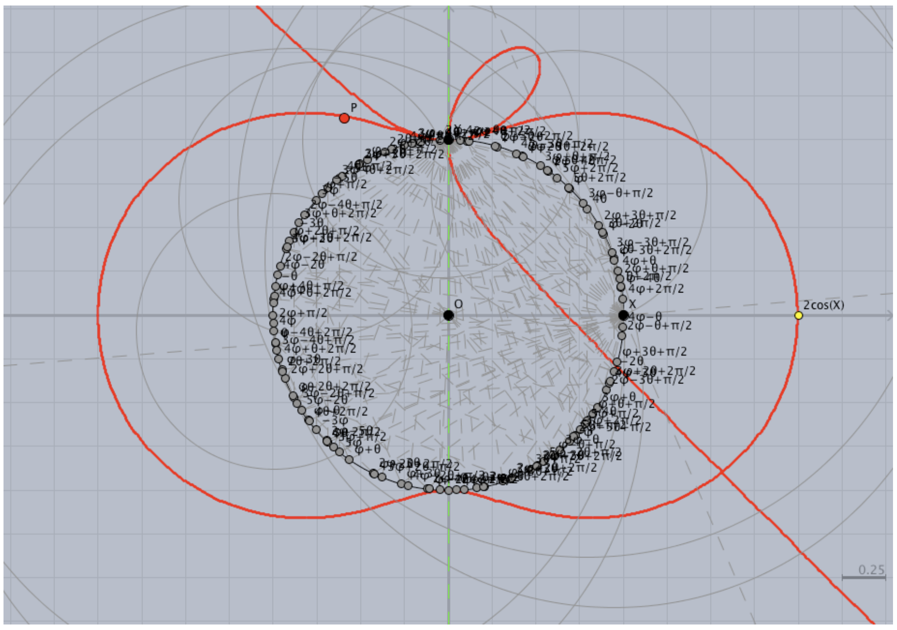
The complexity of our constructions exceeds the $O(d^2)$ bound proven in section 3.3, since we have to deal with the geometric equivalent of rational arithmetics. Simple curves with one-digit coefficients up to degree three usually stay below 500 CindyScript instructions. The apple-shaped curve in figure 4.12 (c) requires 3604 operations without translation of the straight line, and a randomly polynomial of degree eight with integer coefficients in the range $\{-12, \ldots, 12\}$ takes about 12000 primitives, without the translation, too. Anything even more complicated exceeds the usual memory limits



(a) The smooth elliptic curve $y^2 = x^3 - x^2$



(b) A quartic



(c) A curve of degree seven, taken from the *Xalci* gallery [Alg]

Figure 4.12: Some complicated examples

of Java's virtual machine; we do not have to mention that the graphical output takes far too long to be of more than theoretical interest.

4.5 Conclusion

While we have been able to give a proof-of-concept implementation, we have to admit that the examination of the results is no fun so far. The interface to *Cinderella*, while providing a comprehensive choice of geometric operations, is just not designed for killer applications like Kempe linkages. However, there is reason to hope that our implementation will motivate others to use the said interface, and thus invite the *Cinderella* authors to refine it. Even if our project will not benefit, others certainly will.

Still, no dynamical geometry system existing by now allows to perform the task of moving S to trace out the curve, which is the more interesting direction of the translation. Currently we are not aware of any solution to this. Gao et al. mention their software *Geometry Expert* to perform this task, but apparently it is lost, and there is no interest in reanimating it.

Finally, further investigations can be done in the field of lower bounds of exact linkage complexity for particular classes of curves, which still is a widely uncharted terrain.

Bibliography

[ABD08] Timothy Good Abbott, Reid W. Barton, and Erik D. Demaine. Generalizations of Kempe's Universality Theorem. Master's thesis, Massachusetts Institute of Technology, June 2008.

[Alg] Algorithms and Complexity working group at Max-Planck-Institute for Computer Science. Xalci. http://exacus.mpi-inf.mpg.de/cgi-bin/xalci.cgi.

[Eme07] Pavel Emeliyanenko. Visualization of Points and Segments of Real Algebraic Plane Curves. Master's thesis, Saarland University, Saarbrücken, 2007.

[Gaw03] Thomas Gawlick. Über die Mächtigkeit dynamischer Konstruktionen mit verschiedenen Werkzeugen. `http://www.uni-bielefeld.de/idm/service/bib_dok/publikationen/occpap.html`, February 2003.

[GZCG02] Xiao-Shan Gao, Chang-Cai Zhu, Shang-Ching Chou, and Jian-Xin Ge. Automated Generation of Kempe Linkages for Algebraic Curves and Surfaces. In *Mechanism and Machine Theory*, volume 36, pages 1019–1033, 2002.

[JSc] JScience. http://jscience.org/.

[Kem76] Alfred Bray Kempe. On a General Method of describing Plane Curves of the n^{th} degree by Linkwork. In *Proc. of London Mathematical Society*, volume 7, pages 213–216, 1876.

[Kem77] Alfred Bray Kempe. *How to Draw a Straight Line – A lecture on linkages*. Nature Series. Macmillan and Co., London, 1877.

[Ker06] Michael Kerber. Analysis of Real Algebraic Plane Curves. Master's thesis, Saarland University, Saarbrücken, 2006.

[King98] Henry C. King. Configuration Spaces of Linkages in $\mathbb{R}^n$. `arXiv.org:math/9811138`, 1998.

[Kor99] Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. PhD thesis, Swiss Federal Institute of Technology, Zurich, 1999.

[KRG] Ulrich Kortenkamp and Jürgen Richter-Gebert. Cinderella. `http://www.cinderella.de/`.

[Lab08] Oliver Labs. Dynamische Geometrie: Grundlagen und Anwendungen. `http://www.math.uni-sb.de/ag/schreyer/`, 2007–2008.