Decision Complexity in Dynamic Geometry

Ulrich Kortenkamp[1] and Jürgen Richter-Gebert[2]

[1] Institut für Informatik, Freie Universität Berlin, Takustr. 9, 14195 Berlin, Germany, kortenkamp@inf.fu-berlin.de
[2] Institut für Theoretische Informatik, ETH Zürich, ETH Zentrum IFW, 8092 Zürich, Switzerland, richter@inf.ethz.ch

Abstract. Geometric straight-line programs [5, 8] can be used to model geometric constructions and their implicit ambiguities. In this paper we discuss the complexity of deciding whether two instances of the same geometric straight-line program are connected by a continuous path, the *Complex Reachability Problem*.

1 Introduction

Straight-line programs and randomized techniques for proving their equivalence did find their application in geometric theorem proving. Using estimates for the degrees of the variables of a multivariate polynomial given by a straight-line program and evaluations for some random samples, we can prove geometric theorems with much less computational effort than usual [2, 14], for example compared to symbolic methods using Gröbner bases.

An apparent drawback of polynomials is that we have to refer to systems of polynomial equations as soon as we want to describe theorems involving circles or conics. Although there are very powerful methods to do theorem proving in these contexts (e.g. Wu's method, see [13, 12]), it is desirable to have a concept like straight-line programs that is able to describe constructive theorems, and is able to model the dynamic aspects of theorems as they occur in dynamic geometry systems. The implementation of one dynamic geometry system [7, 9] caused the definition of *geometric straight-line programs*, which are one way to approach the above issues.

One question that must be settled before we could use techniques similar to the methods of Schwartz and Zippel [10, 6] to prove geometric theorems is the question of (complex) reachability: Can we move one instance of a geometric theorem continuously into another instance? This paper describes first results on the algorithmic complexity of this question.

2 Geometric Straight-Line Programs

Geometric straight-line programs extend the concept of straight-line programs (see the book of Bürgisser et al. [1] for a detailed discussion of straight-line programs). Informally, a straight-line program (SLP) is a sequence of operations (usually addition, multiplication, subtraction, and sometimes division) that operate on a certain input (usually values of some algebra A) or intermediate results from previous operations.

Straight-line programs are important due to the fact that they provide a very compact description of multivariate polynomials (or rational functions, if we allow divisions). The degree of the polynomials can be much higher than the length of the straight-line program (up to exponential).

In [5] it is shown that geometric constructions using points and lines as objects, and meets and joins as operations, are equivalent to straight-line programs over $\mathbb{R}$ or $\mathbb{C}$. In a way this is a consequence of von-Staudt's approach, who has shown that there is a coordinate-free description of projective geometry [3].

As soon as we want to describe constructions that involve ambiguous operations (like Intersection of Circle and Line, Intersection of Circle and Circle, or Angular Bisector of two lines) the concept of straight-line programs fails. Better said, it is not possible to describe constructions with varying input parameters that behave *continuously* using straight-line programs.

Geometric straight-line programs (GSPs) are a way to keep a concise algebraic description even for constructions involving ambiguous operations. The operations of a straight-line program are replaced by relations from a suitable *relational instruction set (RIS)*. The objects can be choosen arbitrarily, as long as they match the relations. In this paper we will deal with the complex numbers $\mathbb{C}$ as objects and the RIS $R := \{+,-,*,\pm\sqrt{\cdot}\}$ only, and we will emphasize this sometimes by calling them *complex GSPs*.

Again, we refer to [5] for a more formal and detailed description. Here we rely on the readers' intuition and introduce geometric straight-line programs using an example.

Example 1 (A GSP on $(\mathbb{C},R)$). Here is a GSP encoding the expression $\pm\sqrt{{z_1}^2+{z_2}^2}$, with two input variables. The negative indices denote input variables, the other ones index the intermediate results. All statements refer to the indices of previous results or input variables.

Index	Statement	Remark
-2	z_2	Input
-1	z_1	Input
0	$*(-1,-1)$	${z_1}^2$
1	$*(-2,-2)$	${z_2}^2$
2	$+(0,1)$	${z_1}^2+{z_2}^2$
3	$\pm\sqrt{\cdot}(2)$	$\pm\sqrt{{z_1}^2+{z_2}^2}$

A fundamental difference between ordinary straight-line programs and GSPs is that we cannot just "run through" the statements of a GSP in order to calculate the expression for a given input. This is due to the fact that the relations can have different valid outputs for the same input. This gives rise to the notion of an *instance* of a GSP, an assignment of the input parameters and all intermediate results that is compatible with the relations.

Example 2 (Instance of a GSP). An instance for the GSP above is given by

Index	Value	Remark
-2	3	Input z_2
-1	4	Input z_1
0	16	${z_1}^2$
1	9	${z_2}^2$
2	25	${z_1}^2+{z_2}^2$
3	-5	$\pm\sqrt{{z_1}^2+{z_2}^2}$

Observe that all but the last value are determined by the input, and there is only one other instance with the same input (where the last value is 5).

Moving GSPs

For polynomials, or straight-line programs, it is easy to speak about dynamic changes of the input parameters. Since the value of all intermediate results of an SLP is determined by the input, we can vary the input parameters and recalculate the polynomial. Of course, the intermediate results *are* polynomials in the input variables, and as such they are analytic functions, in particular *continuous*.

If we want to do the same with GSPs we must specify how to resolve ambiguities. A natural requirement would be that the intermediate results should be continuous functions in the input parameters. A direct consequence is that the intermediate results must be *analytic* [5] in the following way: Let $U := (u_1,\ldots,u_n), V := (v_1,\ldots,v_n) \in \mathbb{C}^n$ be two inputs for a complex GSP, and let $\gamma\colon [0,1] \mapsto \mathbb{C}^n$ be a path from $\gamma(0) = U$ to $\gamma(1) = V$. If we can find instances of the GSP for every $\lambda \in [0,1]$ such that every intermediate result is an analytic function in λ for $\lambda \in (0,1)$ and a continuous function for $\lambda \in [0,1]$, then these instances form an analytic path.

Here are two examples showing the subtilities of analytic paths:

Example 3 (Square Root). Take the complex GSP with one input that has the $\pm\sqrt{\cdot}$-Relation as the one and only statement, and consider the path

$$\begin{aligned}\gamma\colon [0,1] &\mapsto \mathbb{C}\\ \gamma(\lambda) &= e^{2i\pi\lambda}\end{aligned}$$

For each of the two possible choices at $\lambda = 0$ there is a unique assignment of instances for $\lambda \in (0,1]$ to form an analytic path, which is the proper branch of the complex square root function. The value of the square root at $\lambda = 1$ will be the negative of the value at $\lambda = 0$.

We can find this path by doing analytic continuations along γ, and here in this example it is clear that we can do this for all paths avoiding 0 for $\lambda \in (0,1)$, and only these.

Example 4 (Roots of squares). Take the complex GSP with one input z and with two statements, first multiplying the input with itself and then the $\pm\sqrt{\cdot}$-Relation. The first intermediate result, the square of the input, is determined by the input, and since it is a polynomial, it is analytic in the input z, so it is analytic for any analytic function γ.

The second relation can be simplified to either $+z$ or $-z$, but not to the absolute value function $|x|$, since this would destroy analyticity. We do not have to consider a special path to observe this, it holds for any path.

In the second example there is not always a need to avoid the 0 for the square root function, for example for the path $\gamma(\lambda) = 2\lambda - 1$ there are instances that make it analytic. However, in most considerations it will be a good idea to avoid any zeros of square roots, since these are the critical points where singularities can occur.

3 Complex Reachability and Testing of Polynomials

A problem in straight-line program analysis is to decide whether a given straight-line program is equivalent to another one, i.e. whether it describes the same polynomial (or rational function). The algorithmic complexity of this decision problem is unknown, but there exist polynomial-time randomized algorithms [10]. The main obstacle is that we can neither handle the full, symbolic expression for the polynomial, since the coefficients and the degree of the polynomial can be large, nor the evaluation of the straight-line program for sufficiently large numbers, since the coding length for the intermediate results becomes too large.

If we could find an algorithm to test equivalence of straight-line programs efficiently, then their range of application could be extended to efficient encodings of large numbers. It would also be possible to derive efficient deterministic algorithms to prove geometric theorems.

We will now formulate a version of this decision problem which is equivalent to the equivalence testing problem.

> [SLP zero testing] Given a division-free straight-line program Γ over $\mathbb{Q}$ with one input variable. Is the polynomial p encoded by Γ the zero polynomial?

We will show that this problem is at most as hard as deciding whether we can move analytically from one instance of a GSP to another instance of the same GSP that is different at exactly one intermediate result by giving a polynomial transformation from [SLP zero testing] to the following decision problem:

> [Complex Reachability Problem] Given two instances of a complex GSP with one input variable that differ in exactly one intermediate result. Is it possible to move analytically from the first instance to the second?

We will prove the following theorem, with this corollary as an easy consequence:

Corollary 1. *The* [Complex Reachability Problem] *is algorithmically at least as hard as* [SLP zero testing].

Theorem 1. *There is a polynomial transformation of* [SLP zero testing] *to the* [Complex Reachability Problem], *i.e. we can answer an instance of* [SLP zero testing] *by transforming it to an instance of the* [Complex Reachability Problem] *and answering this.*

Proof. First, we have to clarify how we specify an instance of a complex GSP in a polynomial size of the encoding length of the GSP (where the encoding length of the GSP is the number of bits needed to write down all statements of the GSP). We will deal with GSP inputs that have polynomial encoding length, and then we just have to specify for each $\pm\sqrt{\cdot}$-statement which solution we choose. This can be done using one bit for each decision, saying to choose the solution with the smaller or equal angle in the polar coordinate representation of the two possibilities. We denote an instance by writing down all values of the input variables and a $+$ or $-$ for each decision bit.

Observe that there is no need to evaluate the GSP; indeed, we *must not* evaluate the GSP since this could take exponential time.

Having done this, we assume to have an SLP Γ of length n with one input variable z and want to know whether it describes the zero polynomial. Let us refer to the last result, the polynomial, by $p(z)$.

Let M be the largest constant that can be created using a straight-line program Γ_M of length n and with encoding length less or equal to the encoding length of Γ. Using one additional statement we can write a straight-line program Γ_{2M} that evaluates to $2M$. Thus we can transform Γ in polynomial time and space to Γ' which evaluates $p(z)+2M$. Due to the construction of Γ' the value at $z=0$ of Γ' cannot be 0.

Now we add one additional statement to Γ' in order to evaluate $\pm\sqrt{p(z)+2M}$. Let the new GSP be Γ_1. In a similar way, we also create a GSP Γ_2 that evaluates $\pm\sqrt{zp(z)+2M}$. This requires just one additional statement compared to Γ_1

These two GSPs can be used to decide the zeroness of $p(z)$ using the complex reachability decision. Let $(z=0,+)$ be the start instance, and $(z=0,-)$ the end instance for both Γ_1 and Γ_2.

Now we claim that the reachability decision will be "not reachable" for both Γ_1 and Γ_2 if and only if $p(z)$ is the zero polynomial. For the $\Leftarrow$ direction we observe that $p(z)+2M=2M$ and $zp(z)+2M=2M$, i.e. the arguments of the $\pm\sqrt{\cdot}$-statement are constant and non-zero. So they can never change continuously from one sign decision to the other.

For the $\Rightarrow$ direction we note that $p(z)+2M$ and $zp(z)+2M$ are two polynomials of even and odd resp. odd and even degree if $p(z)\not\equiv 0$. This means that at least one of them has a root of odd multiplicity at, say, z_0. But this means that we can change the sign of the square root by following a path from $z=0$ to $z=z_0+\varepsilon$, cycling once around z_0 and going back from $z=z_0+\varepsilon$ to $z=0$. So for at least one of Γ_1 and Γ_2 the reachability decision will be "reachable." □

4 Remarks

The paper "Randomized Zero Testing of Radical Expressions and Elementary Geometry Theorem Proving" by Daniela Tulone, Chee Yap and Chen Li, that was also presented at ADG 2000, also introduces square roots for straight-line programs. The main difference between our two approaches is that we rely on the implicit sign decision for our notion of geometric theorems, which is different from the usual notion of theorems given by polynomial equations for hypothesis, non-degeneracies and conclusions. Also,

since we only work with complex numbers, we cannot state theorems that are given by semi-algebraic varieties.

Nevertheless, it seems that both the results of both papers can be combined in one or the other way, which we will try to do in our further investigations.

Kurt Mehlhorn pointed out that our transformation shows not only that the complex reachability problem is as hard as to find out whether a polynomial is the zero polynomial, but also as hard as to find out whether a polynomial has at least one root of odd degree.

References

1. Peter Bürgisser, Michael Clausen, and M. Amin Shokrollahi. *Algebraic Complexity Theory*, volume 315 of *A Series of Comprehensive Studies in Mathematics*, chapter 4, pages 103–124. Springer-Verlag, Berlin Heidelberg New York, 1997.
2. Mike Deng. The parallel numerical method of proving the constructive geometric theorem. *Chinese Science Bulletin*, 34:1066–1070, 1989.
3. Hans Freudenthal. The impact of von Staudt's foundations of geometry. In R. S. Cohen, J. J. Stachel, and M. W. Wartofsky, editors, *For Dirk Struik*, pages 189–200. D. Reidel, Dordrecht-Holland, 1974. An article emphasizing the foundation-laying contribution (in terms of purely algebraic description) of von Staudt to projective geometry.
4. Erich Kaltofen. Greatest common divisors of polynomials given by straight-line programs. *Journal of the Association for Computing Machinery*, 35(1):231–264, January 1988.
5. Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. Dissertation, ETH Zürich, October 1999.
6. Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*, chapter 7. Cambridge University Press, Cambridge, 1995.
7. Jürgen Richter-Gebert and Ulrich Kortenkamp. *The Interactive Geometry Software Cinderella*. Springer-Verlag, Heidelberg, 1999.
8. Jürgen Richter-Gebert and Ulrich Kortenkamp. Complexity issues in Dynamic Geometry. In *Proceedings of the Smale Fest 2000*, Hongkong, 2000.
9. Jürgen Richter-Gebert and Ulrich Kortenkamp. *Die interaktive Geometriesoftware Cinderella*. HEUREKA-Klett Softwareverlag, 2000.
10. Jacob T. Schwartz. Probabilistic algorithms for verification of polynomial identities.
11. Volker Strassen. Berechnung und Programm I. *Acta Informatica*, 1:320–335, 1972.
12. Wen-tsün Wu. On the decision problem and the mechanization of theorem-proving in elementary geometry. *Contemp. Math.*, 29:213–234, 1984.
13. Wen-tsün Wu. *Mechanical Theorem Proving in Geometries. Basic Principles. Transl. from the Chinese by Xiaofan Jin and Dongming Wang.* Texts and Monographs in Symbolic Computation. Springer-Verlag, Wien, 1994.
14. Jingzhong Zhang, Lu Yang, and Mike Deng. The parallel numerical method of mechanical theorem proving. *Theoretical Computer Science*, 74:253–271, 1990.