The Third International Workshop on Automated Deduction in Geometry
Zurich, Switzerland, September 25–27, 2000

Decision Complexity in Dynamic Geometry Extended Abstract

Ulrich Kortenkamp and Jürgen Richter-Gebert

[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 [6, 10] 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 expected number of roots in a random sample of evaluations of a polynomial we can prove geometric theorems with much less computational effort than usual [3, 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 [1, 13]), 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 [9, 8] 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 [11, 7] 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 [2] for a detailed discussion of straight-line programs). Infor-

mally, 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 straight-line program (up to doubly exponential).

In [6] 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 [4].

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 [6] 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).

2.1 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* [6] 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 [11]. 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 the corollary as an easy consequence.

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.*

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

References

1. 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.
2. 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. Straight-line programs in algebraic complexity theory.
3. Mike Deng. The parallel numerical method of proving the constructive geometric theorem. *Chinese Sci. Bull.*, 34:1066–1070, 1989.
4. 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.
5. 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.
6. Ulrich Kortenkamp. *Foundations of Dynamic Geometry*. Dissertation, ETH Zürich, 1999.
7. Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*, chapter 7. Cambridge University Press, Cambridge, 1995.
8. Jürgen Richter-Gebert and Ulrich Kortenkamp. *Die interaktive Geometriesoftware Cinderella*. Book & CD-ROM, HEUREKA-Klett Softwareverlag, Stuttgart, 1999.
9. Jürgen Richter-Gebert and Ulrich Kortenkamp. *The Interactive Geometry Software Cinderella*. Book & CD-ROM, Springer-Verlag, Berlin Heidelberg New York, 1999.
10. Jürgen Richter-Gebert and Ulrich Kortenkamp. Complexity issues in dynamic geometry. *In preparation*, 2000.
11. Jacob T. Schwartz. Probabilistic algorithms for verification of polynomial identities. *Symbolic and Algebraic Computation, EUROSAM '79, Int. Symp., Marseille 1979, Lect. Notes Comput. Sci. 72, pages 200–215, 1979.*
12. Volker Strassen. Berechnung und Programm I. *Acta Informatica*, 1:320–335, 1972.
13. Wen-tsün Wu. On the decision problem and the mechanization of theorem-proving in elementary geometry. *Contemp. Math. 29*, pages 213–234, 1984.
14. Jingzhong Zhang, Lu Yang, and Mike Deng. The parallel numerical method of mechanical theorem proving. *Theoretical Computer Science*, 74:253–271, 1990.