The 2D Ray Tracing Problem Using ABCD Lenses and Mirrors Is Turing Complete

Rosemary U. Adejoh
Faculty of Media, Bauhaus-Universität Weimar, Germany

Andreas Jakoby
Faculty of Media, Bauhaus-Universität Weimar, Germany

Sneha Mohanty
Computer Networks and Telematics, University of Freiburg, Germany

Christian Schindelhauer
Computer Networks and Telematics, University of Freiburg, Germany

Abstract

We establish that the two-dimensional ray tracing problem with thin lenses and plane mirrors is Turing-complete, thereby resolving an open question posed by Reif et al. in 1994 as to whether three-dimensional space is necessary for computational universality in optical systems. To this end, we consider the standard approximation of reflection and refraction, namely the ABCD model for paraxial optics, which describes ray propagation through lenses (refraction) via a 2×2 matrix, combined with the geometric reflection model for plane mirrors.

In the absence of mirrors, two-dimensional ray tracing using any combination of lenses in this ABCD matrix model can be described by a single 2×2 matrix–vector product, where the matrix has real entries and determinant 1. Conversely, we show that any such matrix with determinant 1 can be represented as a composition of exactly three appropriately spaced thin lenses. When mirrors are combined with lenses, the ray tracing problem can be described by a flowchart using only two variables, which establishes Turing computability for rational-valued inputs, spaces and matrix entries. Building on this observation, we present a construction of ray tracing that simulates a reversible Turing machine.

We begin with a restricted version of the reversible flowchart problem, in which only two variables and certain linear functions are permitted. We prove that this restricted variant is Turing-complete. We then show that such a flowchart admits a geometric realization using lenses and mirrors in our model, thereby establishing the main result: Turing-completeness of the two-dimensional ray tracing problem with ABCD-model lenses and mirrors.

2012 ACM Subject Classification Theory of computation → Turing machines

Keywords and phrases Turing completeness, optical computation, ray tracing, ABCD matrix, thin lenses, plane mirrors, reversible Turing machine, flowchart, reversible flowchart

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.91

Related Version *Full Version*: `https://arxiv.org/abs/2606.24218` [2]

1 Introduction and Motivation

The computational power of physical systems has long been studied in theoretical computer science as a means of understanding the relationship between computation and the laws of physics. Optical models are particularly attractive in this context: the propagation of light through lenses and its reflection by mirrors naturally implement geometric transformations that can be exploited for computation. Several such models have been shown to simulate universal computation, demonstrating the surprising computational expressiveness of simple optical components.

© Rosemary U. Adejoh, Andreas Jakoby, Sneha Mohanty, and Christian Schindelhauer;
licensed under Creative Commons License CC-BY 4.0

Here, we concentrate on the computational complexity of ray tracing in 2D optical systems, where a light ray enters a maze of lenses and mirrors and we are concerned with the question of where the ray leaves this system. Our main tool is the ABCD ray transfer matrix, which originates from paraxial optics, describing the propagation of light rays through optical elements.

In this model, a ray is characterized by two parameters, i.e. its height offset y (the perpendicular distance from a reference axis) and its angular offset θ, the angle relative to that axis. The propagation of a ray through any optical element – whether a thin lens or a stretch of empty space – is described by multiplication by a 2×2 real matrix:

$$\begin{pmatrix} y_1 \\ \theta_1 \end{pmatrix} = \begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot \begin{pmatrix} y_0 \\ \theta_0 \end{pmatrix} \tag{1}$$

where the matrix has unit determinant ($AD - BC = 1$), i.e. the ABCD matrix is in the special linear group $\mathrm{SL}(2, \mathbb{R})$, which is for 2×2 matrices symplectic [4]. This ABCD matrix formalism, introduced by Kogelnik [8] building on work of Fresnel, is a standard tool in optical engineering for designing telescopes, microscopes, and laser cavities. We demonstrate that it is also a natural framework for universal computation.

1.1 Prior Work and Dimensionality Question

The question of whether optical or mechanical reflection systems can perform universal computation has a rich history. In their seminal paper, Reif et al. [13] proved that three-dimensional ray tracing with mirrors is Turing-complete by showing how to simulate a reversible two-counter automaton. Their construction relies heavily on parabolic mirrors in 3D space to implement the counter operations. Prior to that, Moore [10] established similar results for frictionless billiard ball systems, showing that a single ball in a 3D maze of mirrors can simulate a two-counter machine.

It remains an open question whether three spatial dimensions are necessary for Turing completeness, or do two-dimensional ray tracing suffice.

Recently in [1], we made progress on this question by introducing the ray particle tracing problem, a two-dimensional model involving a ray particle navigating through stationary walls, moving walls, and one-way gates. We showed that this model can simulate a two-stack pushdown automaton (equivalent to a Turing machine) by encoding one stack in the spatial offset of the ray particle's position and the other stack in the temporal offset of its arrival time. While this establishes Turing-completeness in a geometric sense, the model is not purely two-dimensional, i.e. the use of time offset as a computational resource effectively introduces a third parameter, making it a 2.5-dimensional construction. Furthermore, the reliance on moving walls and one-way gates places the model outside the domain of classical static optics.

1.2 Our Contribution

We resolve the dimensionality question discussed above, i.e. we show that the Turing-completeness of the 2D ray tracing problem can be realised using only thin lenses and plane mirrors, thereby getting rid of moving walls (time offset manipulation related components) as well as one-way gates previously used in [1]. This means that only static optical components suffice for our constructions. Also, instead of encoding stacks in spatial offset and time offset, we encode the left and right sides of a 1-tape RTM using the height offset y and the angular offset θ of the ray.

We first investigate the computational power of thin lenses in the ABCD model and show that every matrix with determinant 1 can be realised using a constant number of lenses. By adding mirrors to the model, the ray tracing problem can be expressed as a reversible flowchart that operates on only two variables and uses solely linear operations, such as; *Step*, *Test*, and *Assertion*. This establishes computability for rational-valued inputs. We then show that such restricted reversible flowcharts are reversible Turing complete. This serves as the main tool for proving our central result: that 2D ray tracing is Turing-complete.

The remainder of our paper is organized as follows: Section 2 presents related work on computational models, the reversible flowchart problem as well as ray/beam propagation through complex optical systems. Section 3 establishes our problem definition. Section 4 introduces the abstract components required for Turing-completeness and Theorem 5, our main result on the Turing-completeness of the 2D ray tracing problem, as well as the Turing-completeness of the related two-variable linear reversible flowchart. The proof for this flowchart problem is given in Section 5. Section 6 develops the ABCD matrix formalism and our proof that every ABCD matrix can be constructed with a constant number of thin lenses of the same focal length $f > 0$. Finally, Section 7 combines all components and elaborates on the main result on the Turing-completeness of 2D ray tracing. Section 8 contains the conclusion and open problems. The full version of the paper [2] contains all omitted proofs for Section 6 as well as some figures moved there due to space limitations.

2 Related Work

The theory of reversible computation plays an important role in this paper, which was investigated by Toffoli in [14] based on invertible primitives and invertibility preserving composition rules. Given such constraints, a closer correspondence between the behavior of abstract computing systems and underlying microscopic physical laws can be attained. One of the tools we use in our main result are reversible flowcharts, the fundamentals of which are presented by Yokoyama, Axelsen and Glück in [15]. They show that reversible flowcharts, in its structured and unstructured form are reversible Turing complete, implying that they can compute exactly all injective computable functions. In [3], Axelsen and Glück discuss that Reversible computing is the study of computation models that illustrate both forward and backward determinism. They stress that the property of reversibility should be the starting point of a computational theory of reversible computing. They show that the RTMs can compute exactly all injective, computable functions and that r-Turing completeness is the gold standard for computability in reversible computation models. Chardonnet, Lemonnier and Valiron in [5] show how one can encode a reversible Turing machine using the Theseus language. They build a robust categorical semantics based on join inverse categories and present additional structures to capture pattern-matching and to interpret inductive types as well as recursion.

The work that is closest to our findings is by Miranda and Ramos [9], who show that two-dimensional billiard systems are Turing complete by encoding their dynamics within the framework of Topological Kleene Field Theory. They construct a smooth billiard wall with a fractal zig–zag profile. Their results establish the existence of undecidable trajectories in physically natural two-dimensional billiard-type models. González-Prieto, Miranda and Peralta-Salas [7] present a survey in which they review recent work introducing novel perspectives on the representation of computability through dynamical systems, ranging from classical dynamical universality to modern notions of Turing universality in fluid dynamics and Topological Kleene field theories.

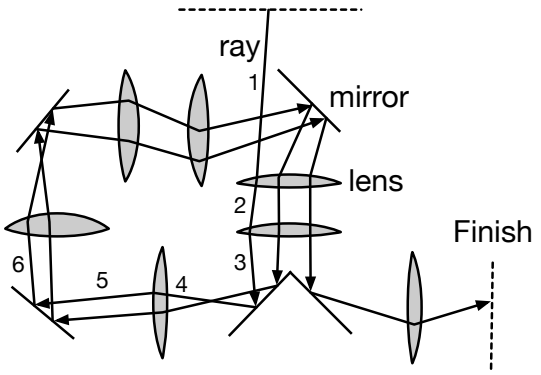

Figure 1 The 2D Ray Tracing problem with lenses and mirrors. The transformations steps of the ray are numbered. The remaining steps follow a similar sequence.

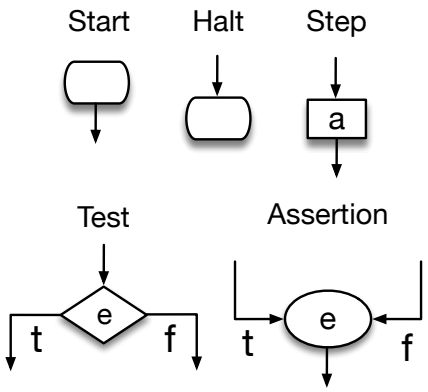

Figure 2 The Reversible Flow Chart Components.

While the ABCD model is only an approximation, it is still relevant in modern optical physics. In [4], Bastiaans and Alieva propose a classification of first-order optical systems based on the eigenvalues of the ray transformation (ABCD) matrix. They present a classification of 1 and 2-dimensional real symplectic matrices based on the distribution of eigenvalues and on if the matrix can be diagonalized. In [6], Corcovilos presents an improved 3×3 matrix method for calculating paraxial ray traces of optical systems that is applicable to arrangements on an optical table, including lenses and mirrors in arbitrary orientation and position. Yura and Hanson [16] describe a formulation of light beam propagation through complex optical systems, including effects such as finite apertures, random jitter, tilt as well as atmospheric turbulence.

3 Problem Definition

We are concerned with the ray tracing problem as depicted in Figure 1.

▶ **Definition 1** (2D ray tracing problem with ABCD lenses and mirrors). *Given a set of mirrors and lenses in two dimensions and a finishing line, the task is to determine whether a ray, given by its initial position and direction, passes the finishing line after being refracted at the lenses according to the ABCD model and reflected at mirrors according to the standard physical law of reflection.*

The law of reflection states that the angle of incidence equals the angle of reflection at the point of contact of the light ray with the mirror. According to the paraxial optics model, we specify the ray by its initial height offset $y_0 \in \mathbb{Q}$ and angular offset $\theta_0 \in \mathbb{Q}$ relative to the optical axis. Note that free space itself influences these parameters, as we will discuss later in Section 6.

For our investigation, we use two reversible computation models. The first is the reversible flowchart model [15]. We use this model to establish Turing computability of all parameters (of the initial ray, the lenses and the mirrors), assuming they are given as rational numbers. Secondly, for the Turing hardness proof we use the reversible Turing machine in quadruple form as discussed in [11].

4 Turing Complete Systems

Recall that a computational model is Turing-complete if it can simulate any given Turing machine. Well-known Turing-complete models include reversible Turing machines, two-stack pushdown automata, and two-counter automata.

In [1] we have shown that the Pinball Wizard problem is Turing-complete. Our simulation uses a network of gadgets which manipulates the arrival time and the arrival position of a ball within a pinball machine and by this we show the Turing-completeness of this problem.

We extend this approach by emphasizing the close relationship to the reversible flowchart [15]. *A reversible flowchart F* is a finite directed graph with three standard types of nodes: *Step*, *Test* and *Assertion*. Here, we have added the single *Start* node and a *Halt* node, indicating the begin and the end of the calculation, see Figure 2.

According to this figure the in- and out-degrees of these nodes are fixed. A further requirement is that the calculation done in a *Step* node, denoted as a, must be invertible, denoted by a^{-1}. In a *Test* node, a condition e is applied in order to determine, which path the calculation takes after this node.

The introduction of the *Assertion* (Join) node is a clever mechanism to ensure invertibility (backward determinism). The designer of a reversible flowchart must ensure that any calculation entering the Assertion node on the input marked as `true` fulfills the condition given by e, while any calculation entering at `false` fulfills the negation $\neg e$.

Similarly, the inverted flowchart can be constructed by inverting the directions of all edges, exchanging Start and Halt nodes, replacing Test with Assertion nodes (and vice versa), and finally replacing the calculation a on the Step nodes with a^{-1}, e.g. the reversal of Figure 5 into Figure 6.

In [15] it has been established that reversible flowcharts with a very basic command set for the Step operations can compute all reversible computations. Here, we use the unstructured reversible flowchart definition, which has been shown to be as expressive as the structured version according to [15].

For analyzing ray tracing we introduce a reduced version of this model.

▶ **Definition 2.** *The 2-Variable Linear Reversible Flowchart [2-LRFC] is a reversible flowchart with the following restrictions*

1. *There are only two variables $y, \theta \in \mathbb{Q}$.*
2. *In each step the calculations a is a linear transformation, i.e.*

$$\begin{pmatrix} y \\ \theta \end{pmatrix} \leftarrow \begin{pmatrix} A & B \\ C & D \end{pmatrix} \begin{pmatrix} y \\ \theta \end{pmatrix} + \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} ,$$

   where the matrix is invertible, i.e. $(AD - BC \neq 0)$, and $A, B, C, D, c_1, c_2 \in \mathbb{Q}$.
3. *The condition e on the Test and Assertions nodes is linear, i.e. $ay + b\theta \leq c$, where $a, b, c \in \mathbb{Q}$.*

Clearly, this model describes only reversible computable functions as the original reversible flowchart model. We show that this model is reversible Turing-complete by simulating a reversible Turing machine in Section 5.

▶ **Theorem 3.** *2-LRFC is reversible Turing-Complete*

For this we give an explicit construction of a reversible Turing machine (RTM), where the Step operations and Test/Assertion conditions are even further reduced. This reduced definition reflects the fact that for ABCD ray tracing the determinant fulfills $AD - BC = 1$ and that mirrors can facilitate Test and Assertion, if we only test for the first variable y.

▶ **Definition 4.** *The Reduced 2-Variable Linear Reversible Flowchart [Red-2-LRFC] is a 2-LRFC with the following restrictions. We only use condition $y < 1$ for the Tests and Assertions and the Step nodes facilitate only one of these calculations:*

1. *Shifting up by one unit:* $\begin{pmatrix} y \\ \theta \end{pmatrix} = \begin{pmatrix} y \\ \theta \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
2. *Shifting down by one unit:* $\begin{pmatrix} y \\ \theta \end{pmatrix} = \begin{pmatrix} y \\ \theta \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix}$
3. Multiplication*:* $\begin{pmatrix} y \\ \theta \end{pmatrix} = M_{mult} \begin{pmatrix} y \\ \theta \end{pmatrix}$*, for* $M_{mult} = \begin{pmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{pmatrix}$
4. Switch*:* $\begin{pmatrix} y \\ \theta \end{pmatrix} = M_{switch} \begin{pmatrix} y \\ \theta \end{pmatrix}$*, for* $M_{switch} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$

We show that Red-2-LRFC is reversible Turing complete by giving an explicit construction of a simulation of a reversible Turing Machine (RTM) in Section 5, which then implies Theorem 3.

Returning to the ray tracing problem in two-dimensional paraxial optics, the variable y denotes the spatial offset from the optical axis of the lenses, and θ denotes the angle of the ray. Neither M_{switch} nor M_{mult} can be directly realized by a single interaction of a ray with a lens or a mirror. Furthermore, while the ray travels between these elements, its offset y changes, while θ remains constant. To familiarize the reader with the basics of this ABCD optical theory, we provide a short introduction in Section 6, where we also discuss which matrix operations can be realized by a constant sequence of lenses without mirrors.

This enables us, in Section 7, to give an explicit construction of lenses and mirrors that transforms the reversible flowchart simulating an RTM from Section 5 into an instance of the two-dimensional ray tracing problem. For this, we place Test and Assertion operations at the nodes of a carefully spaced grid. We then show how the four step operations, together with a connection element implementing the identity function, can be embedded along the edges of this grid. This establishes our main result.

▶ **Theorem 5.** *The two-dimensional ray tracing problem with ABCD lenses and plane mirrors is Turing-complete.*

5 Turing Hardness of the Two-Variable Linear Reversible Flowchart

For our simulation, we use the quadruple form of a reversible Turing machine (RTM) as discussed in [11].

▶ **Definition 6** (Reversible Turing Machine)**.** *A one-tape reversible Turing machine in quadruple form is defined by $M = (Q, \Sigma, q_0, q_f, \delta)$, where Q is a finite set of states, $\Sigma = \{0, 1, \beta\}$ is the tape alphabet, $q_0 \in Q$ is the initial state, and $q_f \in Q$ is the final state. The transition relation is given by*

$$\delta \subseteq (Q \times \Sigma \times \Sigma \times Q) \ \cup \ (Q \times \{\square\} \times \{-1, 0, 1\} \times Q).$$

A transition is either:

- *a **computing transition**, i.e., a transition $[p, x, z, q]$ with states $p, q \in Q$ and symbols $x, z \in \Sigma$. In this case, the head does not move; it reads symbol x and writes symbol z.*
- *a **head-moving transition**, i.e., a transition $[p, \square, d, q]$ with states $p, q \in Q$, a special symbol $\square \notin \Sigma$ (indicating that no symbol is read), and a movement direction $d \in \{-1, 0, +1\}$. In this case, the head moves left (-1), right $(+1)$, or stays in place (0), and the tape remains unchanged.*

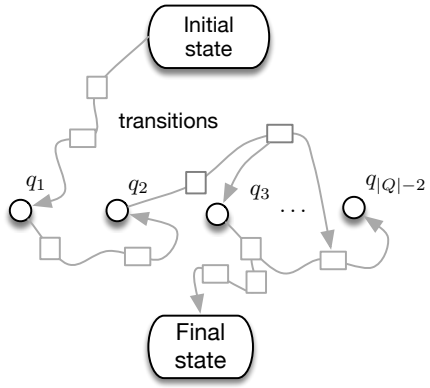

Figure 3 State nodes and transitions of the reversible flowchart simulating an RTM.

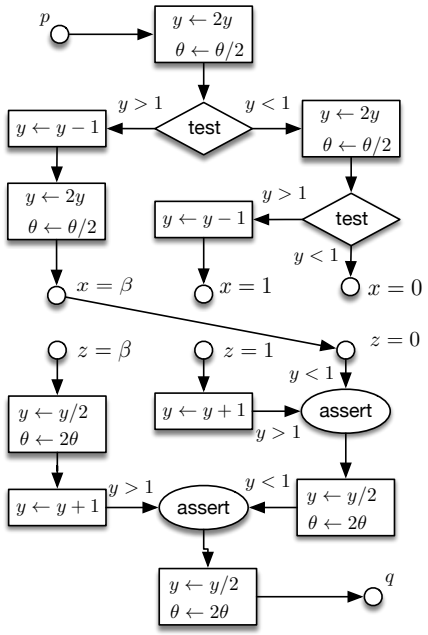

Figure 4 Reversible flowchart for the computation transition (p, x, z, q) for $x = \beta$ and $z = 0$.

To ensure reversibility, the transition relation δ must be injective in both forward and backward directions. That is, for any two distinct transitions $[p_1, x_1, y_1, q_1] \in \delta$ and $[p_2, x_2, y_2, q_2] \in \delta$, the following conditions hold:

- ***Forward determinism:*** *if $p_1 = p_2$, then $x_1 \neq x_2$.*
- ***Backward determinism:*** *if $q_1 = q_2$, then $y_1 \neq y_2$.*

We assume that within a computation the RTM always alternates from step to step between computing and head moving transition.

In the following construction, we ensure that the two flowchart variables are always bounded, i.e., $\theta, y \in \mathbb{Q} \cap (-2, 2)$. This is motivated by the limited size of the lenses used later on. Furthermore, we avoid integer values, i.e., $\theta, y \notin \mathbb{Z}$, since this prevents rays from hitting the boundaries of mirrors.

We now describe how to construct a reduced 2-variable linear reversible flowchart for a given reversible Turing machine M. The initial and final states are represented by the Start node and Halt node of the flowchart, respectively. For the remaining states of the Turing machine, we introduce $|Q| - 2$ state nodes in the flowchart; see Figure 3.

The tape content and head position are encoded by the two variables y and θ, which assume consistent values upon entering each state node. The left part of the tape (relative to the head position) is encoded by θ, while the symbol under the head together with the right part of the tape is encoded by y. Let

$$\ldots s_{-3}\, s_{-2}\, s_{-1}\, s_0\, s_1\, s_2 \ldots$$

be the infinite tape in some state q, where s_0 denotes the symbol under the head. Positive indices correspond to the right side of the tape, and negative indices to the left side. We encode each symbol $s \in \{0, 1, \beta\}$ by an integer from $\{0, 1, 2\}$ via $v(0) = 0$, $v(1) = 1$, and $v(\beta) = 2$. The tape is then represented as follows:

$$y = \sum_{i=0}^{\infty} v(s_i) \cdot 4^{-i-1}, \tag{2}$$

$$\theta = \sum_{i=1}^{\infty} v(s_{-i}) \cdot 4^{-i}. \tag{3}$$

Note that an empty tape $\ldots\beta\beta\beta\ldots$ corresponds to the constant values $y = \theta = \frac{2}{3}$. In particular, neither y nor θ ever attains an integer value. Using this encoding, we can realize the following elementary operations:

1. Reading the symbol s_0 under the head: $v(s_0) = \lfloor 4y \rfloor$.
2. Reading the symbol s_{-1} to the left of the head: $v(s_{-1}) = \lfloor 4\theta \rfloor$.
3. Replacing the symbol s_0 by s_1 at the head position: $y \leftarrow y + \frac{v(s_1) - v(s_0)}{4}$.
4. Moving the head to the right:

$$y \leftarrow 4y - v(s_0), \tag{4}$$

$$\theta \leftarrow \frac{\theta + v(s_0)}{4}. \tag{5}$$

5. Moving the head to the left:

$$y \leftarrow \frac{y + v(s_{-1})}{4}, \tag{6}$$

$$\theta \leftarrow 4\theta - v(s_{-1}). \tag{7}$$

We connect the state nodes of the reversible flowchart using components that implement the transitions given by δ of the RTM.

Given a *computation transition* $[p, x, z, q]$, we construct a sub-flowchart connecting state node p to state node q. Let $v(x) = b_0 + 2b_1$ and $v(z) = b'_0 + 2b'_1$, where $b_i, b'_i \in \{0, 1\}$.

We build a depth-two decision tree. First, we apply a multiplication step to test the most significant bit b_1 of $v(x)$. If $b_1 = 1$, we remove it by a suitable shift. We then test for b_0 in the second layer. After these steps, the symbol x is effectively erased from the head position.

To write the new symbol z, we use the division operation

$$M_{\text{div}} := M_{\text{switch}} \cdot M_{\text{mult}} \cdot M_{\text{switch}}^3 = M_{\text{mult}}^{-1} = \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & 2 \end{pmatrix},$$

which can be realized by a constant sequence of operations. We first add 1 to y if $b'_0 = 1$, then apply the division, add b'_1 to y, and finally apply another division; see Figure 4.

For a *head-moving transition* $[p, \square, d, q]$, if $d = 0$, we simply connect state node p to q.

If $d = -1$, we move the head to the left. To do so, we extract the symbol s_{-1} from the left side of the tape. We first apply a switch operation exchanging θ and y, and then construct a decision tree as above to determine $v(s_{-1}) = b_0 + 2b_1$ and remove it from y. We then apply the inverse switch operation, which can be realized as

$$M_{\text{switch}}^{-1} = M_{\text{switch}}^3 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$

Finally, we add the extracted bits to y; see Figure 5.

The remaining case $d = 1$ corresponds to a right move of the head and is obtained by inverting the construction for the left move; see Figure 6.

This completes the construction and proves Theorem 3, since we simulate an RTM by a reduced 2-variable linear reversible flowchart.

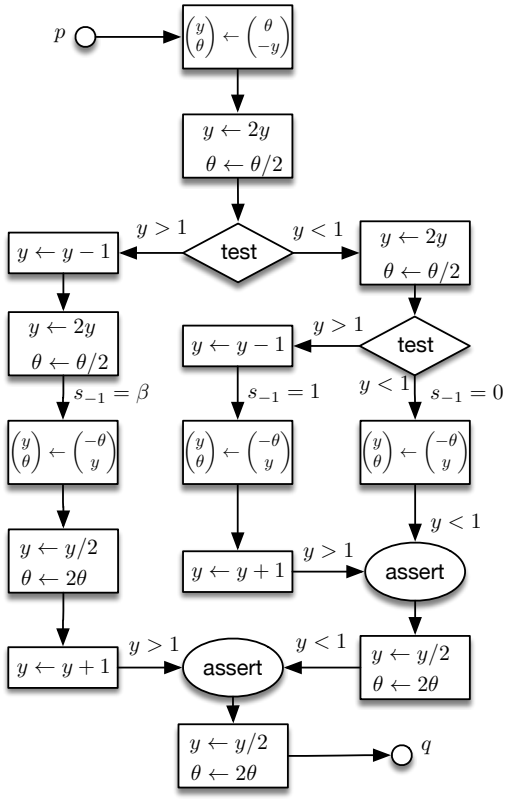

Figure 5 Left head movement $[p, \square, -1, q]$.

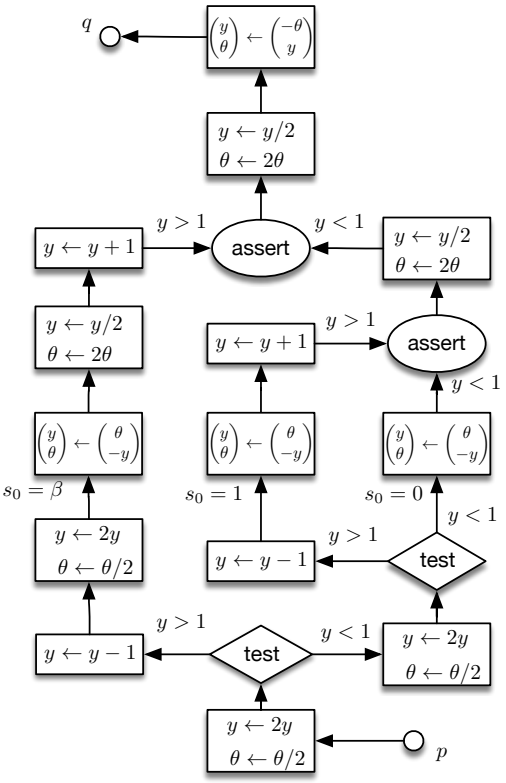

Figure 6 Right head movement $[p, \square, +1, q]$.

6 The ABCD Ray Transfer Model

We give a brief introduction to the relevant aspects of the ABCD model in optics, which is used to describe lenses and curved mirrors in two dimensions. For a more detailed treatment, we refer to standard optics textbooks such as [12].

In this model, a reference x-axis is fixed along which optical elements, such as lenses or curved mirrors, are placed. At a position x_0 on this axis, a light ray is characterized by its offset y_0 (the perpendicular distance from the axis) and its angle θ_0 relative to the axis. This state is represented by the vector (y_0, θ_0). As the ray propagates through the system, it is refracted by lenses, reflected by mirrors, and travels along straight lines between these elements. At a later position x_1, we observe the exiting ray and measure its offset and angle, denoted by (y_1, θ_1), again with respect to the x-axis.

Let us first consider propagation through a region of free space of length d, so that $x_1 = x_0 + d$. In this case, the angle remains unchanged, i.e., $\theta_1 = \theta_0$, while the offset satisfies $y_1 = y_0 + d \sin\theta_0$. The ABCD model assumes small angles, allowing the approximation $\sin\theta_0 \approx \theta_0$. Using this approximation, propagation through free space can be described by a *space ABCD matrix* S_d, which acts via matrix multiplication; see Figure 8.

$$\begin{pmatrix} y_1 \\ \theta_1 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix}}_{S_d} \cdot \begin{pmatrix} y_0 \\ \theta_0 \end{pmatrix} \tag{8}$$

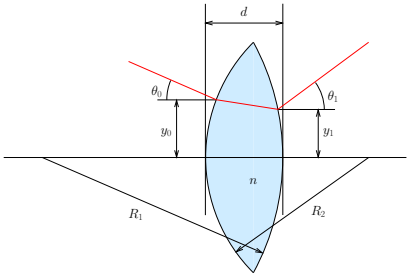

Figure 7 The geometry of a thin lens. Note that we assume $d \to 0$.

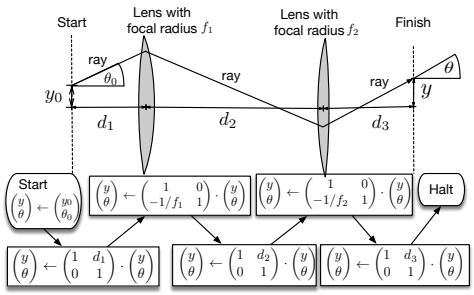

Figure 8 The computation induced by two lenses with focal points f_1 and f_2 as flowchart.

Without going into further detail similar approximating matrices can be derived for thin lenses, thick lenses, concave and convex mirrors, which all can be described by Equation (1), where $A, B, C, D \in \mathbb{R}$. In all cases, the resulting matrices have unit determinant, i.e.,

$$\begin{vmatrix} A & B \\ C & D \end{vmatrix} = 1 \ . \tag{9}$$

Thus, in addition to the space matrix S_d, the *thin lens ABCD matrix* T_f is relevant for our considerations.

$$T_f = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \qquad \text{where} \qquad \frac{1}{f} = (n-1) \cdot \left(\frac{1}{R_1} - \frac{1}{R_2} \right) \ . \tag{10}$$

The value $f \in \mathbb{R} \setminus \{0\}$ describes the *focal length* according to the *lens maker's formula*. We assume that the diameter of this lens can be neglected. Here, n describes the refractive index of the glass and we assume that the refractive of air to be 1. Note that any value $f \in \mathbb{R} \setminus \{0\}$ can be achieved by an appropriate choice of R_1 and R_2, where R_1 and R_2 are the radii of the two sides of the lens, respectively, see Figure 7.

We first investigate, whether all real-valued unit determinant matrices can be the result of ray propagating through thin lenses and space. It is well known [12] that for all ABCD matrices with $C \neq 0$ there exist *principal planes* in distances p_1 before and p_2 after it, which reduce it to a thin lens:

$$S_{p_2} \cdot \begin{pmatrix} A & B \\ C & D \end{pmatrix} \cdot S_{p_1} = T_f \qquad \text{for } f = \frac{-1}{C}, \quad p_1 = \frac{1-D}{C}, \quad p_2 = \frac{1-A}{C} \ . \tag{11}$$

Clearly this equation makes only sense if $p_1, p_2 > 0$ which is not the case for all values of A, B, C, D.

So, the following theorem answers the question, whether it is possible to construct every unit determinant matrix with a constant number of single thin lenses.

▶ **Theorem 7.** *Every rational-valued matrix with unit determinant can be constructed with exactly three thin lenses of appropriately chosen focal lengths and using appropriate spacing. There are matrices where less than three lenses are not enough.*

In the proof, given in full detail in the corresponding technical report [2], we have adapted the focal lengths to the given spaces, while in reality, the type of lenses is given and one adjusts the spacing. So, we wonder, how many *different types* of lenses are necessary to construct all ABCD matrices. In fact a single type of lens with given focal length f is enough.

Clearly for any focal length $f < 0$ not all matrices can be generated from such a concave lens, since its corresponding ABCD matrix and the space matrix S_d have only non-negative entries. Therefore, no matrix with negative entries, like $M_{f'}$ with $f' > 0$, can be produced by matrix multiplication.

But for focal length $f > 0$, any such single thin lens is universal.

▶ **Theorem 8.** *Every rational-valued matrix with determinant 1 can be produced with a constant number of thin lenses of a given rational focal length $f > 0$ and adjusted distances.*

The detailed proof is given in [2]. The number of necessary thin lenses is at most 18. We are aware that a construction with a smaller number of thin lenses exists. The exact upper and lower bound for this number is left for future work. The following theorem shows that a constant number of lenses also can fill a large enough distance d.

▶ **Theorem 9.** *Given a single type of thin lenses with focal length $f > 0$, then for every matrix with determinant 1 there is some d_0 such that for all $d \geq d_0$ there is a construction with constant number of lenses which length from start to finish is exactly d.*

The proof of this Theorem can be found in [2].

7 From ABCD Matrices to 2D Ray Tracing Turing Completeness

We now construct an optical system based on ABCD lenses and mirrors that simulates an RTM. We start from the reduced 2-variable linear reversible flowchart described in Section 5. The ray is guided through a grid with cell size g; see Figure 9. The ray starts at a grid node, and the Halt node is also represented by a grid node. Grid nodes are either left empty or equipped with mirrors tilted at 45°. Empty grid nodes are used for the Start and Halt nodes, as well as for perpendicular crossings. The mirrors redirect an incoming ray with angle θ to an outgoing ray with angle $\pi/2 - \theta$, either to the left or to the right. We also use mirrors to implement Test and Assertion operations, as shown in Figure 11.

Edges connecting grid nodes implement the Step operations *shift-up*, *shift-down*, *multiplication*, *switch*, or a *wire*. The wire corresponds to the identity function, which cannot be realized by empty space alone, since this would result in multiplication by the space matrix S_g. We use four rotated versions of each component to accommodate rays traveling in the four cardinal directions.

The operations of the reversible flowchart can be implemented using mirrors and lenses modeled by ABCD matrices as follows. We begin with the Step operations, which must match the grid spacing. In Theorem 9, we showed that for any fixed lens with focal length $f > 0$, one can choose a grid size g such that a construction of exactly this length exists. However, this general result cannot be applied directly here, since the resulting lens positions may be irrational, as they arise from solving quadratic equations. Since for the hardness result it suffices to pick one instance, we choose (among many possibilities) the following combinations resulting in three components of overall lengths 9 and 12.

$$M_{\text{mult}} = S_1 \cdot T_1 \cdot S_{5/2} \cdot T_1 \cdot S_4 \cdot T_1 \cdot S_{3/2} \tag{12}$$

$$M_{\text{switch}} = S_3 \cdot T_1 \cdot S_3 \cdot T_1 \cdot S_4 \cdot T_1 \cdot S_2 \tag{13}$$

$$M_{\text{wire}} = S_1 \cdot T_1 \cdot S_3 \cdot T_1 \cdot S_3 \cdot T_1 \cdot S_2 \ . \tag{14}$$

A remaining issue arises from the choice of $y, \theta \in (-2, 2)$, which may cause the edges to become too wide. As a consequence, lenses may interfere with other parallel edges. Moreover, the underlying assumption of the ABCD model, namely $\sin\theta \approx \theta$, is clearly

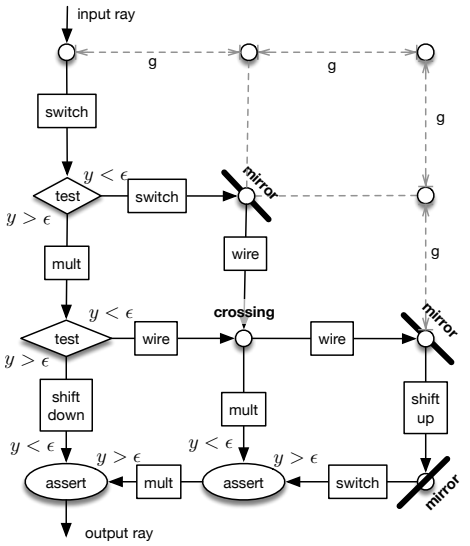

Figure 9 Geometry of the grid-based layout for the 2D ray tracing problem.

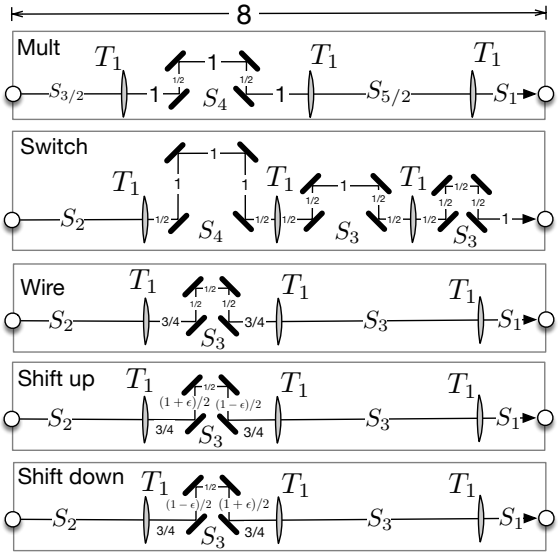

Figure 10 Step operations implemented along the edges of the grid.

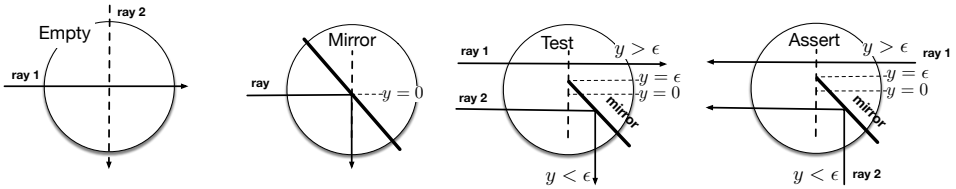

Figure 11 Crossing/connection, redirection, Test, and Assertion at the nodes of the ray-tracing grid.

violated. We resolve this problem by scaling the interval to $(-2\epsilon, 2\epsilon)$ for some rational $\epsilon > 0$, for example $\epsilon = \frac{1}{100}$, and replacing the tape encoding by $y = \epsilon \sum_{i=0}^{\infty} v(s_i) \cdot 4^{-i-1}$, and $\theta = \epsilon \sum_{i=1}^{\infty} v(s_{-i}) \cdot 4^{-i}$. Accordingly, we replace all Test and Assert conditions by $y < \epsilon$, and change the shift operations to $y \leftarrow y + \epsilon$ (*shift-up*) and $y \leftarrow y - \epsilon$ (*shift-down*), without changing anything else.

In order to obtain a smaller grid size $g = 8$, we fold all three mirror lens systems using a periscope mirror construction; see Figure 10. This periscope is adapted to transform the wire into *shift-up* and *shift-down* operations, which moves the ray respectively up or down by ϵ. This yields the edge operations shown in Figure 10 and the node operations in Figure 11. Note that the matrix multiplication operations remain unchanged, since the transformation corresponds to a uniform scaling of all values.

Summarizing, to prove the hardness of the 2D ray tracing problem, we start from a given reversible Turing machine and construct the corresponding reduced 2-variable linear reversible flowchart as described in Section 5. We then embed this flowchart into a grid of size g, placing Start, Halt, Test, and Assert nodes at grid points such that all connections can be routed along separate edges. Crossings and arbitrarily long wires are realized using empty nodes. Next, we apply the scaling with a rational $\epsilon > 0$, e.g., $\epsilon = 1/100$, and replace each grid node by the corresponding mirror component for Test, Assertion or redirection. Finally, we substitute each edge operation by the corresponding three-lens construction as described in Equations (12)–(14). The correctness of the construction follows from the implementations shown in Figure 10 and Figure 11.

Hence, given an input ray with offset y encoding the initial tape content and an empty tape encoded in θ, the output ray encodes the final tape configuration via its offset and angle, provided that the RTM halts. Moreover, the constructed system reaches the Halt node if and only if the simulated RTM halts. This establishes the main result: two-dimensional ray tracing with ABCD-modeled lenses and mirrors is Turing-complete.

8 Conclusions and Open Problems

We have shown that the two-dimensional ray tracing problem with thin lenses and plane mirrors is Turing-complete, thereby resolving the question of whether full three-dimensional space is necessary for computational universality in optical systems. Our construction demonstrates that the classical ABCD matrix formalism from paraxial optics provides a natural and intuitive framework for universal computation. The main tool in our simulation of a reversible Turing machine is the close relationship between ray tracing and the concept of reversible flowcharts. We show that such flowcharts are Turing-complete even when restricted to two rational variables and linear functions. In combination with a detailed analysis of the capabilities of ABCD-modeled lenses, this yields our main result: the Turing-completeness of the two-dimensional ray tracing problem with ABCD-modeled lenses and mirrors.

Unlike the ray particle tracing model introduced in [1], which simulates a two-stack automaton and encodes one of the stacks in the arrival time (thereby introducing a 2.5-dimensional construction), our model operates entirely within the two-dimensional plane. Furthermore, our construction uses only thin lenses and plane mirrors, i.e., stationary components. In particular, we eliminate the need for moving walls (used for time-offset manipulation), one-way gates, or parabolic mirrors. It is worth noting that the constructions in [1] could be simplified by simulating a reversible Turing machine. In that case, one-way gates would no longer be required.

As an open problem, we ask how many lenses of a given focal length $f > 0$ are necessary to realize an arbitrary matrix with unit determinant. Clearly, our bound of 18 lenses can be improved. It also remains open whether a lens system with rational parameters can be constructed for sufficiently large rational total length (from start to finish). Moreover, a construction using concave and convex mirrors in place of lenses appears feasible within the ABCD model. Finally, the ultimate question is whether Turing-completeness of the two dimensional ray tracing problem can be established for natural optical systems that do not rely on approximations.

References

1 Rosemary U. Adejoh, Andreas Jakoby, Sneha Mohanty, and Christian Schindelhauer. How Pinball Wizards Simulate a Turing Machine. In C. Aiswarya, Ruta Mehta, and Subhajit Roy, editors, *45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025)*, volume 360 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSTTCS.2025.4.

2 Rosemary U. Adejoh, Andreas Jakoby, Sneha Mohanty, and Christian Schindelhauer. The 2d ray tracing problem using abcd lenses and mirrors is turing complete, 2026. arXiv:2606.24218.

3 Holger Bock Axelsen and Robert Glück. What do reversible programs compute? In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures*, pages 42–56, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-19805-2_4.

4 Martin J. Bastiaans and Tatiana Alieva. Classification of lossless first-order optical systems and the linear canonical transformation. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 24 4:1053–62, 2007. URL: https://api.semanticscholar.org/CorpusID:23969043.

5 Kostia Chardonnet, Louis Lemonnier, and Benoît Valiron. Semantics for a Turing-Complete Reversible Programming Language with Inductive Types. In Jakob Rehof, editor, *9th International Conference on Formal Structures for Computation and Deduction (FSCD 2024)*, volume 299 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 19:1–19:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.FSCD.2024.19.

6 Theodore A. Corcovilos. Beyond the abcds: A better matrix method for geometric optics by using homogeneous coordinates. *American Journal of Physics*, 91(6):449–457, June 2023. doi:10.1119/5.0083069.

7 Ángel González-Prieto, Eva Miranda, and Daniel Peralta-Salas. Universality in computable dynamical systems: old and new. *Journal of Physics: Complexity*, 6(3):035014, September 2025. doi:10.1088/2632-072X/ae00b5.

8 Herwig Kogelnik. Imaging of optical modes—resonators with internal lenses. *Bell System Technical Journal*, 44(3):455–494, 1965. doi:10.1002/j.1538-7305.1965.tb01672.x.

9 Eva Miranda and Isaac Ramos. Classical billiards can compute, 2025. doi:10.48550/arXiv.2512.19156.

10 Cristopher Moore. Unpredictability and undecidability in dynamical systems. *Physical Review Letters*, 64(20):2354, 1990. doi:10.1103/PhysRevLett.64.2354.

11 Kenichi Morita. Reversible turing machines. In *Theory of Reversible Computing*, pages 103–156. Springer, 2017. doi:10.1007/978-4-431-56606-9_5.

12 Justin Peatross and Michael Ware. *Physics of light and optics*. Brigham Young University, Department of Physics Provo, UT 84602, 2015.

13 John H. Reif, J. Doug Tygar, and A. Yoshida. Computability and complexity of ray tracing. *Discrete & Computational Geometry*, 11:265–288, 1994. doi:10.1007/BF02574009.

14 Tommaso Toffoli. Reversible computing. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, pages 632–644, Berlin, Heidelberg, 1980. Springer Berlin Heidelberg. doi:10.1007/3-540-10003-2_104.

15 Tetsuo Yokoyama, Holger Bock Axelsen, and Robert Glück. Fundamentals of reversible flowchart languages. *Theoretical Computer Science*, 611:87–115, 2016. doi:10.1016/j.tcs.2015.07.046.

16 H. T. Yura and S. G. Hanson. Optical beam wave propagation through complex optical systems. *J. Opt. Soc. Am. A*, 4(10):1931–1948, October 1987. doi:10.1364/JOSAA.4.001931.