1 Existential Theory of the Real Numbers

The results of the previous chapters have shown why an analysis of $\mathbf{NP}_{\mathbb{R}}$-complete problems is important. However, there is one major point missing in order to justify the relevance of the completeness concept: We have to establish that all problems in class $\mathbf{NP}_{\mathbb{R}}$ actually can be decided by a BSS algorithm.

The definition of class $\mathbf{P}_{\mathbb{R}}$ directly implies any problem in $\mathbf{P}_{\mathbb{R}}$ to be decidable. If it would turn out that the same does not hold true for $\mathbf{NP}_{\mathbb{R}}$, then the meaning of Theorem Theorem **??** would be significantly reduced. In that case we would only have established that (polynomially) decidable problems are easier to be solved than undecidable problems - a trivial fact which would immediately imply as well $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$.

In the present chapter we therefore want to present an algorithm showing that the 4-Feasibility problem, and thus any problem in class $\mathbf{NP}_{\mathbb{R}}$, is decidable. As a consequence, this result substantiates the importance of the $\mathbf{P}_{\mathbb{R}} = \mathbf{NP}_{\mathbb{R}}$?-question. Recall that the same result in the Turing model was obtained in a straightforward manner. The 3-Satisfiability problem is decidable in simple exponential time just by a brute search among all possible assignments for the given formula. Since the number of assignments for a given, fixed number of Boolean variables is finite, this algorithm will terminate. In contrast, for a polynomial $f \in \mathbb{R}[\underline{x}]$ in n real variables we cannot plug in all of the uncountable many assignments for the variables into f and check, whether one of it evaluates to 0. The guessing space is uncountable and an obvious algorithm for deciding the question is not at hand.

As we will see in this chapter the corresponding question has a long history tracing back to work done by Tarski in the 30s of the last century. He was the first who gave a decision procedure (which can be formalized as a BSS algorithm) for the so-called existential theory over the real numbers. His famous theorem is settled in a (more general) model-theoretic framework. It deals with the question whether *real-closed fields* allow *quantifier elimination* for *first-order formulas* .

We want to present Tarski's algorithm and use it to derive a decision procedure for all problems in $\mathbf{NP}_{\mathbb{R}}$. Unfortunately, the complexity of the algorithm is tremendous; it does not imply a simple exponential time bound for the 4-Feasibility problem in the BSS model. Much work in the recent 25 years was devoted to speed up Tarski's procedure, see [14, 76, 28, 29, 57, 4, 3]. After a huge amount of research the currently fastest algorithms (at least from the theoretical side) for the solution of problems in $\mathbf{NP}_{\mathbb{R}}$ provide a sim-

ple exponential running time, thus meeting the bounds known for discrete complexity theory and the class **NP**.
EVENTUELL: AT THE END OF THIS CHAPTER WE SHALL ALSO REPRESENT ONE OF THESE FASTER ALGORITHMS GIVEN BY RENEGAR [57].

In order to give an idea of the question quantifier elimination is dealing with let us start with an easy example.

Example 1.0.1 Consider a univariate polynomial $f(x) := x^2 + a \cdot x + b$ of degree two. In high school one studies the solution behavior of the equation $f(x) = 0$, i.e. the pair (a, b) was given and depending on the values it is figured out whether the equation is solvable or not.
In a more formal way the problem can be stated as follows: Given a formula

$$\psi(a, b) \equiv \exists\, x \in \mathbb{R}\; x^2 + a \cdot x + b = 0$$

with free variables a, b we want to figure out conditions on a and b under which the formula is true. The well-known answer tells us that we should consider another formula, namely

$$\rho(a, b) \equiv a^2 - 4 \cdot b \geq 0$$

and that ρ is equivalent to ψ, i.e. $\rho(a, b) \Leftrightarrow \psi(a, b)\ \forall\, a, b \in \mathbb{R}$. The advantage of considering ρ instead of ϕ is that no more quantifiers appear in ρ. Given a pair (a, b) we just plug it into ρ, evaluate the left hand-side and check, whether it is non-negative.
Note that the above situation completely changes if we require a, b and a solution x to be rational numbers. □

The above example provides an easy one of a quantifier elimination. Given a formula with quantifiers, we want to know whether there exists an equivalent formula which is quantifier free. Moreover, we want to compute the former from the letter (by a BSS algorithm). This is the general task of quantifier elimination. Tarski's theorem states that such an elimination is possible (in the sense of: there exists an equivalent quantifier free formula and it is computable) for formulas over the real numbers, which just contain the arithmetic operations, the order, existential and universal quantifiers over the reals and the logical connectives (i.e. formulas in first-order logic, see below).

As references for our presentation of Tarski's method we point the reader to [9], [31] and the unpublished paper [50], which is based on an idea by Mushin.

1.1 The univariate case: Sturm's Theorem

Because of its importance in the proof of Tarski's theorem, and because of its historical interest we start with a presentation of Sturm's theorem, [64]. It is devoted to compute the number of real zeros a univariate real polynomial has. The first basic definition is

Definition 1.1.1 Let $a := (a_0, \ldots, a_d) \in \mathbb{R}^{d+1}$; the number $SC(a)$ of *sign changes* in a is defined as

$$SC(a) \ := \ |\{(i, i+k) \mid a_i \cdot a_{i+k} < 0 \text{ and } a_{i+r} = 0 \ \forall \ 0 < r < k, k \geq 1\} \ .$$

□

For example, $SC(-1, 0, -1, 2, 0, -2) = 2$.

Definition 1.1.2 Let $p \in \mathbb{R}[x]$ be a polynomial in one real variable x and let $a < b \in \mathbb{R}$. A *Sturm chain* of p in $[a, b]$ is a finite sequence $(f_0, f_1, \ldots, f_s)$ of univariate polynomials in $\mathbb{R}[x]$ such that the following conditions are satisfied:

i) $f_0 = p$;

ii) the polynomial f_s has no real roots in $[a, b]$;

iii) for $0 < i < s$ and $\alpha \in [a, b]$ such that $f_i(\alpha) = 0$ it is

$$f_{i-1}(\alpha) \cdot f_{i+1}(\alpha) < 0;$$

iv) for $\alpha \in (a, b]$ such that $f_0(\alpha) = 0$ it is

$$\begin{array}{l} (f_0 \cdot f_1)(\alpha - \epsilon) < 0 \\ (f_0 \cdot f_1)(\alpha + \epsilon) > 0 \end{array}$$

for sufficiently small values of $\epsilon > 0$.

□

Note that condition iii) implies f_i and f_{i+1} to have no common zeros in $[a, b]$.

The first theorem in this section states that a Sturm chain can be used to count the number of real zeros of a univariate polynomial in an interval. Thereafter, we will show how a Sturm chain can be obtained for an arbitrary polynomial.

Theorem 1.1.3 Let $(f_0, \ldots, f_s)$ be a Sturm chain for a polynomial $p \in R[x]$ on $[a, b], a < b$. We denote by $SC(x)$ the number $SC(f_0(x), \ldots, f_s(x))$ for $x \in \mathbb{R}$. Then $SC(a) - SC(b)$ is the number of different real zeros of p in $(a, b]$.

Proof. The proof proceeds by bookkeeping the changes in $SC(x)$ when x is moved from a to b. Consider an $\alpha \in [a, b]$. There are three cases:

1.) If $f_i(\alpha) \neq 0$ for all $0 \leq i \leq s$ then $SC(x)$ is constant in a neighborhood of α.

2.) If $f_0(\alpha) \neq 0$ and $f_i(\alpha) = 0$ for an $0 < i < s$, then condition iii) of the definition of a Sturm chain implies that the signs of f_{i-1} and of f_{i+1} are constant in a neighborhood of α. For example, we might have a pattern like

	$f_{i-1}(x)$	$f_i(x)$	$f_{i+1}(x)$
$\alpha - \epsilon < x < \alpha$	$+$		$-$
$x = \alpha$	$+$	0	$-$
$\alpha < x < \alpha + \epsilon$	$+$		$-$

for $\epsilon > 0$ small. We see that the value $SC(x)$ remains constant for $x \in (\alpha - \epsilon, \alpha + \epsilon)$, no matter what the signs of $f_i(x)$ are for $x < \alpha$ or $x > \alpha$.

3.) If $f_0(\alpha) = 0$ for $\alpha > a$ then the value $SC(x)$ is reduced by 1 if x passes α from the left to the right. This is true because of condition iv) in the definition of a Sturm chain.

For example, suppose

$$sgn(f_0(x)) = \begin{cases} -1 & \alpha - \epsilon < x < \alpha \\ 1 & \alpha < x < \alpha + \epsilon \end{cases} .$$

This implies

$$sgn(f_1(x)) = \begin{cases} 1 & \alpha - \epsilon < x < \alpha \\ 1 & \alpha < x < \alpha + \epsilon \end{cases} .$$

The other cases can be treated similarly.

Note that for $\alpha = a$ the value $SC(x)$ is locally constant in $[a, a + \epsilon]$. ■

Corollary 1.1.4 With the above notation the number of different real zeros of a polynomial $p \in \mathbb{R}[a]$ is given by $SC(-\infty) - SC(\infty)$.

Proof. Since for any univariate polynomial q we have $\lim_{|x|\to\infty} q(x) = \pm\infty$, there is an $M \in \mathbb{R}$ such that the sign pattern of a Sturm chain for p evaluated in $x < -M$ will be the same as the one for $-M$ and the sign pattern of the same Sturm chain evaluated for $x > M$ will be the same as for argument M. Thus, the number we are looking for is given by $SC(-M) - SC(M)$ by the previous theorem. ■

The final problem is to find a Sturm chain for a given polynomial p.

Theorem 1.1.5 Let $p \in \mathbb{R}[x]$ be a polynomial without multiple roots in $[a,b], a < b$. Put $f_0 := p, f_1 := p'$ and define polynomials $f_i, 2 \leq i \leq s$ via

$$f_{i-2} = f_{i-1} \cdot g_i - f_i \ ,$$

where $deg(f_i) < deg(f_{i-1})$ and s is minimal such that $f_s(x) \neq 0 \ \forall \ x \in [a,b]$. Then $(f_0, \ldots, f_s)$ is a Sturm chain for p. It can be computed by the Euclidean Algorithm.

Proof. We have to check conditions i) till iv) of Definition 1.1.2. The first two conditions are obvious from the fact that p has no multiple zeros.
Ad iii) Let $f_i(x) = 0$ for an index $0 < i < s$ and an $x \in [a,b]$.
The construction implies

$$f_{i-1}(x) = -f_{i+1}(x) \text{ resp. } f_{i-1}(x) \cdot f_{i+1}(x) \leq 0 \ .$$

If the latter product would vanish we obtain $f_0(x) = f_1(x) = 0$ by the formulas defining the f_i's , i.e. p would have multiple zeros.
Ad iv) Let $f_0(x) = 0$, then x is a local minimum for $f_0^2(x)$ and we can conclude

$$(2 \cdot f_0 \cdot f_1)(x - \epsilon) < 0 \text{ and } (2 \cdot f_0 \cdot f_1)(x + \epsilon) > 0$$

for $\epsilon > 0$ sufficiently small. ■

It might be difficult to check whether $f_s(x) \neq 0$ on $[a,b]$. In order to avoid this test we can continue with Euclidean's Algorithm until we have computed a greatest common divisor of p and p'. It is easy to see that the potentially longer list of polynomials again gives a Sturm chain for p. If p has multiple zeros, one can first compute the greatest common divisor $\tilde{p}$ of p and p' and then apply the above algorithm to $\frac{p}{\tilde{p}}$. If we want to count multiplicities as

well, then the algorithm has also to be performed for $\tilde{p}$. Finally, note that the Euclidean algorithm is used by taking negative rests; the reason for this variant becomes obvious in the proof.

Example 1.1.6 Let $p(x) := x^2+a\cdot x+b =: f_0(x)$ be an arbitrary polynomial of degree 2. We obtain $f_1(x) := p'(x) = 2 \cdot x + a$ as well as $f_0(x) = f_1(x) \cdot (\frac{x}{2} + \frac{a}{4}) - (\frac{a^2}{4} - b)$. The latter implies $f_2(x) := \frac{a^2}{4} - b$. The sign behavior for (f_0, f_1, f_2) is given by

$$\begin{array}{rccc} x & sgn(f_0) & sgn(f_1) & sgn(f_2) \\ -\infty & + & - & sgn(a^2-4b) \\ \infty & + & + & sgn(a^2-4b)\ . \end{array}$$

which results in

$$\begin{array}{l} SC(-\infty) - SC(\infty) = 0 \ \Leftrightarrow\ a^2 - 4\cdot b < 0 \\ SC(-\infty) - SC(\infty) = 1 \ \Leftrightarrow\ a^2 - 4\cdot b = 0 \\ SC(-\infty) - SC(\infty) = 2 \ \Leftrightarrow\ a^2 - 4\cdot b > 0\ . \end{array}$$

□

1.2 Tarski's Theorem

We are now going to generalize the ideas behind Sturm's Theorem to the multivariate setting and several polynomials. We shall consider a finite family $\mathcal{P} := \{p_1, \ldots, p_s\}$ of polynomials depending on variables $y_1, \ldots, y_n, x$. Since the proof of Tarski's Theorem is based on a stepwise elimination of variables, we divide the latter into the block $y := (y_1, \ldots, y_n)$ and the single variable x. The aim is to show how questions about the sign vectors (see definition below) of the system $\mathcal{P}$ can be reduced to questions about the sign vectors of another family of polynomials, this time only depending on variables y (thus, x is eliminated).
The proof, though not extremely difficult, is divided into several parts whose interaction is a bit complex.

For a moment let us deal again with univariate polynomials and define a particular table related to a system of such polynomials.

Definition 1.2.1 Let $p_1, \ldots, p_s$ denote polynomials in one variable.

a) $w := (w_1, \ldots, w_s) \in \{-1, 0, 1\}^s$ is a *valid sign vector* for the family $(p_1, \ldots, p_s)$ iff there is a $x \in \mathbb{R}$ such that $sgn\ p_i(x) = w_i\ \forall 1 \leq i \leq s$. Here, we define the sign function as usual by

$$sgn\ x := \begin{cases} 1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} .$$

b) Let $x_1 < x_2 < \ldots < x_N$ be all points in $\mathbb{R}$ in which at least one of the p_i vanishes. Define $x_0 := -\infty$, $x_{N+1} := \infty$ as well as $I_k := (x_k, x_{k+1})$ for $k = 0, \ldots, N$.

Then the *sign table* $SGN(\mathcal{P}) = SGN(p_1, \ldots, p_s)$ of $\mathcal{P}$ is defined as

$$SGN(p_1,\ldots,p_s) := \left(\begin{pmatrix} sgn\ p_1(I_0) \\ \vdots \\ sgn\ p_s(I_0) \end{pmatrix}, \begin{pmatrix} sgn\ p_1(x_1) \\ \vdots \\ sgn\ p_s(x_1) \end{pmatrix}, \begin{pmatrix} sgn\ p_1(I_1) \\ \vdots \\ sgn\ p_s(I_1) \end{pmatrix}, \right.$$
$$\left. \ldots, \begin{pmatrix} sgn\ p_1(x_N) \\ \vdots \\ sgn\ p_s(x_N) \end{pmatrix}, \begin{pmatrix} sgn\ p_1(I_N) \\ \vdots \\ sgn\ p_s(I_N) \end{pmatrix} \right) .$$

□

Remark 1.2.2 a) The value $sgn\ p_j(I_k)$ denotes $sgn\ p_j(x)$ for a point $x \in I_k$. This notation is independent of the particular x because all p_j have a constant sign in the interval I_k.
b) $SGN(p_1, \ldots, p_s)$ contains all valid sign vectors of the system $\mathcal{P}$. A sign vector can appear several times in the table: if $w \in \{-1, 0, 1\}^s$ is attained in $\widehat{x}$ and $\tilde{x}$, then w has to be listed at least twice if the system attains another sign vector for a point $x \in (\widehat{x}, \tilde{x})$.
c) The sets $I_0, \{x_1\}, I_1, \{x_2\}, \ldots, I_N, \{x_N\}, I_{N+1}$ give precise information about the connected components of $\mathbb{R}$, on which the system $(p_1, \ldots, p_s)$ does not change its sign vector. They are called the *sign consistent connected components* of $\mathcal{P}$. □

Example 1.2.3 Consider $\mathcal{P} := \{p_1, p_2, p_3\}$, where $p_1(x) := x^3 - 1, p_2(x) := 2 \cdot x + 2, p_3(x) := 1$. The points $x_1 := -1$ and $x_2 := 1$ are the zeros of one of the p_i. We obtain a table with 3 columns and 5 rows, given as

$$SGN(p_1, p_2, p_3) := \left(\begin{pmatrix} -1 \\ -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right) .$$

□

If the p_i are polynomials in several variables y, x, a similar table can easily be defined for a fixed choice of $c \in \mathbb{R}^n$ for the variables y. Write a p_i as polynomial in x with coefficients in $\mathbb{R}[y]$, i.e. as an element in $\mathbb{R}[y][x]$.

Definition 1.2.4 Let $\mathcal{P}$ be as above and let $c \in \mathbb{R}^n$ be fixed. We denote by $SGN(\mathcal{P}, c)$ the sign table obtained for the univariate family of polynomials $p_1(c, \bullet), \ldots, p_s(c, \bullet)$. □

Before we continue with the description of Tarski's algorithm let us outline the general ideas it is following. Suppose we want to decide whether a multivariate polynomial family $p_1, \ldots, p_s$ has a common zero (any other requirement on the signs of the p_i either in one single point or in all points is allowed as well). Then there exists a vector $(c, \bar{x}) \in \mathbb{R}^{n+1}$ such that $p_i(c, \bar{x}) = 0 \ \forall \ 1 \leq i \leq s$. Therefore, if we consider the table $SGN(\mathcal{P}, c)$ it contains a column of zeros. And vice versa: if for a particular choice of c the table $SGN(\mathcal{P}, c)$ contains a column of zeros, then the polynomials in $\mathcal{P}$ have a common zero. Hence, the question of deciding whether a family of polynomials has a common zero is equivalent to the question of deciding whether there is a choice $c \in \mathbb{R}^n$ such that $SGN(\mathcal{P}, c)$ has a zero column. The main idea of the decision procedure is the way of approaching the latter question. In a first step, starting from $\mathcal{P}$ another finite family $BC(\mathcal{P})$ of polynomials is constructed; the elements of $BC(\mathcal{P})$ are polynomials depending on y only, i.e. $BC(\mathcal{P}) \subset \mathbb{R}[y]$. It then turns out (and this is the crucial fact) that for fixed $c \in \mathbb{R}^n$ the signs of the polynomials of $BC(\mathcal{P})$ evaluated in c already determine the table $SGN(\mathcal{P}, c)$. The latter transformation moreover can be computed by a BSS algorithm (which was, of course, not Tarski's formulation). Note that even though there are uncountably many choices for c, the number of different sign patterns given by $BC(\mathcal{P})$ is finite.
We can then figure out those patterns which produce a zero column in $SGN(\mathcal{P}, c)$ and continue our procedure recursively by deciding, whether these sign patterns for the family $BC(\mathcal{P})$ can be realized. The latter question depends only on the variables y, i.e. x has been eliminated.

We now describe the construction of the above mentioned family $BC(\mathcal{P})$. It is defined in two steps; first, we build the closure $C(\mathcal{P})$ of the initially given family $\mathcal{P} := \{p_1, \ldots, p_s\} \subset \mathbb{R}[y][x]$ under certain operations. Then, $BC(\mathcal{P})$ is obtained as the intersection of $C(\mathcal{P})$ with $\mathbb{R}[y]$.
The operations are very much related to the ideas behind the proof of Sturm's theorem. Let $f(x) := \sum_{i=0}^{d} a_i \cdot x^i \in \mathbb{R}[y][x]$ be a polynomial with $a_i \in \mathbb{R}[y], a_d \neq 0 \in \mathbb{R}[y]$ and $d \geq 1$.

We define

(1) *Derivative* : $F_1(f) := \frac{\partial f}{\partial x} \in \mathbb{R}[y][x]$;

(2) *Leading coefficient* : $F_2(f) := a_d \in \mathbb{R}[y]$;

(3) *Omitting leading coefficient* : $F_3(f) := f - F_2(f) \cdot x^d \in \mathbb{R}[y][x]$;

The fourth operation F_4 is applied to a pair $f(x) := \sum_{i=0}^{d} a_i \cdot x^i$ and $g(x) := \sum_{k=0}^{e} b_k \cdot x^k$ of polynomials in $\mathbb{R}[y][x]$, where $d \geq e \geq 1$ and $f \neq g$. It is defined as

(4) *Modified remainder* : $F_4(f,g) := r \in \mathbb{R}[y][x]$. Here, r is obtained by dividing f by g with rest as polynomials in x and multiplying the resulting equation by the common denominator $b_e^{d-e+1} \in \mathbb{R}[y]$. That is, r is the unique polynomial in $\mathbb{R}[y][x]$ given by the equation

$$b_e^{d+e-1} \cdot f(x) = g(x) \cdot h(x) + r(x) \ , \ deg_x(r) < deg_x(g), h \in \mathbb{R}[y][x].$$

Remark 1.2.5 The reason for the multiplication with b_e^{d+e-1} in (4) is that we want to set up an iterative algorithm later on. Therefore, after eliminating x we have to start anew, this time with a family of polynomials in y (and not of rational functions). □

Definition 1.2.6 Let $\mathcal{P} := \{p_1, \ldots, p_s\}$ be a finite family of polynomials in $\mathbb{R}[y][x]$. The *closure* $C(\mathcal{P})$ of $\mathcal{P}$ is defined to be the (finite) set of polynomials in $\mathbb{R}[y][x]$ satisfying the following conditions:

i) $\mathcal{P} \subseteq C(\mathcal{P})$;

ii) $\forall\ p \in C(\mathcal{P})$ such that $deg_x(p) \geq 1$ the polynomials $F_i(p), 1 \leq i \leq 3$ belong to $C(\mathcal{P})$;

iii) $\forall\ p, q \in C(\mathcal{P})$ such that $deg_x(p) \geq deg_x(q) \geq 1$ the polynomial $F_4(p,q)$ belongs to $C(\mathcal{P})$.

□

Proposition 1.2.7 Let $\mathcal{P}$ be as above, then $C(\mathcal{P})$ is well defined, finite and computable within a finite number of steps by a BSS machine on input $\mathcal{P}$.

Proof. Starting from $\mathcal{P}$ we enlarge $\mathcal{P}$ step by step by applying one of the operations $F_1, \ldots, F_4$ either to elements of $\mathcal{P}$ or to elements already derived previously during this process. Note that every F_i results in a polynomial in $\mathbb{R}[y][x]$ which has a smaller degree with respect to x than its argument (resp. its arguments in case of F_4). Thus, since $\mathcal{P}$ is finite and since all operations can only be applied to polynomials of degree at least 1, the closure $C(\mathcal{P})$ is constructed after a finite number of steps (all of which are clearly computable by a BSS machine). ■

The polynomials among $C(\mathcal{P})$ we are mainly interested in are those which depend on the variables y only.

Definition 1.2.8 For $\mathcal{P}$ as above the *base* of the closure $C(\mathcal{P})$ is defined as

$$BC(\mathcal{P}) \ := \ \mathbb{R}[y] \cap C(\mathcal{P}).$$

□

Example 1.2.9 Let $\mathcal{P} := \{p\}$ be given by $p(y, x) := x^2 \cdot y - 1$ (y a single variable). We can construct $C(\mathcal{P})$ as

$$\begin{array}{lcl} q_1(y,x) & = & F_1(p) = 2 \cdot x \cdot y \\ q_2(y,x) & = & F_2(p) = y \\ q_3(y,x) & = & F_3(p) = p - F_2(p) \cdot x^2 = -1 \\ q_4(y,x) & = & F_4(p, q_1) = -1 \\ q_5(y,x) & = & F_1(q_1) = 2 \cdot y \\ q_6(y,x) & = & F_2(q_1) = 2 \cdot y \\ q_7(y,x) & = & F_3(q_1) = 0 \end{array}$$

Since the F_i have to be applied to polynomials of degree ≥ 1 w.r.t. x, at this point no more polynomials can be obtained. The closure is computed. Finally, $BC(\mathcal{P})$ is given by $\{q_2, q_3, q_4, q_5, q_6, q_7\}$. Even though some of the polynomials appear twice it is important to have them all included. As we will see later on, the point here is to know which operation has been used in order to obtain a particular polynomial, and not whether some operations yield the same result. The reader might get a feeling about the tremendous complexity of computing $C(\mathcal{P})$ for more complex systems already from this trivial example. □

Theorem 1.2.10 Let $\mathcal{P} = \{p_1, \ldots, p_s\} \subset \mathbb{R}[y][x]$ and $c \in \mathbb{R}^n$. Then there is a BSS algorithm which, given the signs of all polynomials $q \in BC(\mathcal{P})$ if evaluated in c, computes the table $SGN(\mathcal{P}, c)$.

Proof. The proof is a (reverse) induction on the steps performed in order to compute the closure $C(\mathcal{P})$ starting from $\mathcal{P}$. Consider an arbitrary, finite BSS algorithm $\mathcal{A}$ computing $C(\mathcal{P})$, see Proposition 1.2.7. Let $BC(\mathcal{P}) \cup \{f_1, \ldots, f_\ell\}$ and

$$C(\mathcal{P}) \;=\; BC(\mathcal{P}) \cup \{q_1, \ldots, q_t\} \; .$$

Here, we suppose that the q_i are given in the reverse order in which they are produced by algorithm $\mathcal{A}$; that is, q_1 is the last polynomial in $C(\mathcal{P}) \setminus BC(\mathcal{P})$ produced by $\mathcal{A}, q_2$ is the second last and so on.
In particular, if we consider the polynomials $F_1(q_i), F_2(q_i)$ or $F_3(q_i)$ for a $q_i \in C(\mathcal{P}) \setminus BC(\mathcal{P})$, then they can be found among $BC(\mathcal{P}) \cup \{q_1, \ldots, q_{i-1}\}$ (again, the same polynomial might of course have occurred earlier, compare the remark at the end of Example 1.2.9). Similarly, $F_4(q_i, q) \in BC(\mathcal{P}) \cup \{q_1, \ldots, q_{i-1}\}$ for $q \in \{q_1, \ldots, q_{i-1}\}$ because then q was produced later than q_i and $F_4(q_i, q)$ later than both.

The proof proceeds by induction on $BC(\mathcal{P}) \cup \{q_1, \ldots, q_i\}$ with respect to increasing i, i.e. we show that, given the signs of $BC(\mathcal{P})$ in c, the table $SGN(\{q_1, \ldots, q_i\}, c)$ can be computed from $SGN(\{q_1, \ldots, q_{i-1}\}, c)$. Since $C(\mathcal{P})$ contains $\mathcal{P}$ we will finally obtain $SGN(\mathcal{P}, c)$.
The tables we construct during the algorithm are structured as follows: the rows are named by a polynomial and the columns correspond either to intervals in which all the listed polynomials have a constant sign or to symbolic roots of at least one of the polynomials. Adding a new polynomial in the induction step increases the table by one row for the new polynomial and by as many columns as that polynomial has zeros which were not already among the zeros of the previously listed polynomials. Whereas the ordering of the rows is irrelevant (we add one row at the bottom) the ordering of the columns is not. A new column has to be included into the scheme according to the usual ordering of the real zero related to it in comparison with the other zeros (however, we do not compute such a root, see Remark 1.2.11 below).
Our first scheme for $BC(\mathcal{P})$ has ℓ rows indexed by $f_1, \ldots, f_\ell$ and 3 columns for $x_0 := -\infty, I_1 := (-\infty, \infty)$ and $x_1 := \infty$. Every row has the form

$$sgn(f_j) \quad sgn(f_j) \quad sgn(f_j)$$

due to the fact that for fixed c all $f_j(c)$ are constant real numbers.

For $i = 1$ the polynomial q_1 is the final one produced by algorithm $\mathcal{A}$ such that $deg_x(q_1) \geq 1$. Therefore, $deg_x(q_1) = 1$ because if $deg_x(q_1) > 1$, then q_1

could not be the first polynomial in the above list since the application of F_1 to q_1 would yield another polynomial in $C(\mathcal{P}) \setminus BC(\mathcal{P})$.
So we can represent q_1 as

$$q_1(y, x) = f_j(y) \cdot x + f_k(y)$$

with $f_j, f_k \in BC(\mathcal{P})$. If $f_j(c) = 0$, the new row added for q_1 is just a copy of the row indexed by f_k. If $sgn(f_j(c)) = \epsilon_j \in \{-1, 1\}, q_1$ has one zero $\bar{x}$. Enlarge the sign table by doubling the previous mid-column for $(-\infty, \infty)$ and adding a new column for $\bar{x}$ between the two copies. The five entries for the new bottom row indexed by q_1 are given as

$$-\epsilon_j \quad -\epsilon_j \quad 0 \quad \epsilon_j \quad \epsilon_j \ .$$

In order to perform the induction step we assume that the sign table $SGN(BC(\mathcal{P} \cup \{q_1, \ldots, q_{i-1}\})$ has already been constructed for an $i - 1 \geq 1$. Let q_i be the next polynomial added. The polynomials $F_2(q_i)$ giving the leading coefficient of q_i and $F_3(q_i)$ extracting the rest of q_i are listed in the table. Thus, if $F_2(q_i)(c) = 0$ the column for q_i will be the same as that for $F_3(q_i)$ and we are done.
We therefore assume $F_2(q_i)(c) \neq 0$. Denote by

$$x_1 < x_2 < \ldots < x_m$$

all the real zeros among the polynomials in $\{q_1, \ldots, q_{i-1}\}$ which do not identically vanish as univariate polynomials in x if c is chosen as assignment for the variables y. We have to deal with four different cases:

i) compute the sign of $q_i(c, \bullet)$ if evaluated in $-\infty$ or ∞;

ii) compute the sign of $q_i(c, \bullet)$ if evaluated in one of the $x_k, 1 \leq k \leq m$;

iii) find (with respect to the ordering of the x_k) the new zeros of q_i and

iv) compute the sign of $q_i(c, \bullet)$ if evaluated in the neighboring intervals of the new zeros found in iii).

ad i) Due to $F_2(q_i)(c) \neq 0$, the leading coefficient of $F_1(q_i)$, which is a positive multiple of $F_2(q_i)(c)$, is non-vanishing as well. Therefore,

$$sgn(q_i(c, -\infty)) = -sgn(F_1(q_i)(c))$$

and

$$sgn(q_i(c, \infty)) = sgn(F_1(q_i)(c)).$$

These values are already listed in the initial table.

ad ii) For an $x_k, 1 \leq k \leq m$, consider a polynomial q among $\{q_1, \ldots, q_{i-1}\}$ such that $q(c, x_k) = 0$ and the polynomial $q(c, x)$ does not identically vanish as univariate polynomial in x. Such a q exists by definition of x_k and of $\{q_1, \ldots, q_{i-1}\}$. The polynomial $F_4(q_i, q)$ is listed in the table we started with and satisfies the equation

$$F_2(q)(c)^{deg_x(q_i)-deg_x(q)+1} \cdot q_i(c, x_k) = F_4(q_i, q)(c, x_k) \ .$$

Here, $F_2(q)(c) \neq 0$; moreover, we can determine $sgn(q_i(c, x_k))$ from $sgn(F_2(q)(c)$ and $sgn(F_4(q_i, q)(c, x_k))$, which both are already available.

ad iii) Any root $\bar{x}$ of q_i which is not yet listed, say $\bar{x} \in (x_j, x_{j+1})$ for some $j \in \{0, \ldots, m\}$, is a simple root of $q_i(c, \bullet)$. Otherwise, $\bar{x}$ would be as well a root of $F_1(q)(c)$ and therefore appear among $x_1, \ldots, x_m$.
In the table for $BC(\mathcal{P} \cup \{q_1, \ldots, q_i\})$ three columns are added and replace the one between those columns corresponding to x_j and x_{j+1}. Simplicity of $\bar{x}$ as root for $q_i(c, \bullet)$ implies

$$sgn(q_i(c, x_j)) \cdot sgn(q_i(c, x_{j+1})) = -1 \ .$$

Vice versa, a new root in (x_j, x_{j+1}) only appears if this product is -1. We obtain the three new columns indexed by $(x_j, \bar{x}), \bar{x}$, and $(\bar{x}, x_{j+1})$. The bottom row (corresponding to q_i) in these columns gets the entries

$$sgn(q_i(c, x_j)) \quad 0 \quad sgn(q_i(c, x_{j+1}))$$

which can be computed by step ii).
The update for the other rows indexed by $f_1, \ldots, f_\ell, q_1, \ldots, q_{i-1}$ in the newly added columns are obvious. They are filled either with a zero if the entire previous row was filled with zeros, or they are filled with the unique non-zero sign found in one (or both) of the neighboring columns.

iv) There are some missing entries for the bottom row in those columns corresponding to intervals. If a missing entry appears between two columns already filled with $(1, 1), (1, 0)$ or $(0, 1)$, we add a 1; otherwise, we add a -1.

The construction of $SGN(BC(\mathcal{P} \cup \{q_1, \ldots, q_i\})$ is finished. ■

Remark 1.2.11 It is important to point out that the above algorithm does not compute the zeros of any involved polynomial. The only information used is that of the ordering of such zeros in relation to each other. The zeros are thus only treated in a symbolic manner. □

Example 1.2.12 We continue with Example 1.2.9. The set $BC(\mathcal{P})$ was computed as $\{y, -1, -1, 2y, 2y, 0\}$. The different sign vectors for $BC(\mathcal{P})$ are given by three different choices for y, namely for $y < 0, y = 0$ and $y > 0$. The corresponding sign vectors are

$$\text{for } y < 0: \begin{pmatrix} -1 \\ -1 \\ -1 \\ -1 \\ -1 \\ 0 \end{pmatrix}; \quad \text{for } y = 0: \begin{pmatrix} 0 \\ -1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}; \quad \text{for } y > 0: \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \\ 1 \\ 0 \end{pmatrix}.$$

In the first table these columns are listed with three copies.

The polynomial $p(y, x)$ has a zero (only) for positive y. We want to extract this information by using the above construction for a choice $c > 0$ for the variable y. The initial sign-table is

$-\infty$	$(-\infty, \infty)$	∞	
1	1	1	q_2
-1	-1	-1	q_3
-1	-1	-1	q_4
1	1	1	q_5
1	1	1	q_6
0	0	0	q_7

In the first step, we have to add a row for the polynomial $q_1(y, x) = 2 \cdot y \cdot x$. Both $q_6 = F_2(q_1)$ and $q_7 = F_3(q_1)$ are already listed. Since $sgn(q_6(c)) = 1 > 0$, the polynomial $x \to q_1(c, x)$ has a zero x_2 and the sign of $q_1(c, x)$ is -1 for $x < x_2$ and 1 for $x > x_2$. The algorithm replaces the middle column by three new ones; the new middle column corresponds to x_2. We obtain for $SGN(\{q_2, \ldots, q_7, q_1\}, c)$ the table

$-\infty$	$(-\infty, x_2)$	x_2	(x_2, ∞)	∞	
1	1	1	1	1	q_2
-1	-1	-1	-1	-1	q_3
-1	-1	-1	-1	-1	q_4
1	1	1	1	1	q_5
1	1	1	1	1	q_6
0	0	0	0	0	q_7
-1	-1	0	1	1	q_1

Adding the original p to the table we first note that $F_2(p)(c) = q_2(c) = c > 0$, i.e. p's leading coefficient does not vanish. The value $sgn(p(c, -\infty))$ equals $-sgn(F_1(p(c, -\infty))) = -sgn(q_1(c, -\infty)) = 1$ and $sgn(p(c, \infty)) = sgn(F_1(p(c, \infty))) = sgn(q_1(c, \infty)) = 1$.
For the sign of p in (c, x_2) the algorithm requires to consider the polynomials q_1 (as a polynomial which has x_2 as its zero), $F_4(p, q_1) = -1$ and $F_2(p)(c) = c$. The modified remainder equation in x_2 is

$$c^{2-1+1} \cdot p(c, x_2) = -1$$

and, because of $c^2 > 0$, we get $sgn(p(c, x_2)) = -1$. For the above five columns, we have now computed three entries for p: the first and the fifth column have entry 1 and the third has entry -1. Thus, two new simple roots of p have to be added; a zero $x_1 < x_2$ and a zero $x_3 > x_2$. The sign-table is enlarged by replacing both the old second and the old fourth column by three new ones.
For the first block of these new columns (i.e. for the columns $2, 3$ and 4 in the enlarged table) the row indexed by p gets the entries $1, 0, -1$. For the second block (i.e. for the columns $6, 7$ and 8) it gets the entries -1,0,1. Performing the trivial updates for the other polynomials we obtain the sign-table $SGN(\{q_1, \ldots, q_7, q_1, p\}, c)$ as:

$-\infty$	$(-\infty, x_1)$	x_1	(x_1, x_2)	x_2	(x_2, x_3)	x_3	(x_3, ∞)	∞	
1	1	1	1	1	1	1	1	1	q_2
-1	-1	-1	-1	-1	-1	-1	-1	-1	q_3
-1	-1	-1	-1	-1	-1	-1	-1	-1	q_4
1	1	1	1	1	1	1	1	1	q_5
1	1	1	1	1	1	1	1	1	q_6
0	0	0	0	0	0	0	0	0	q_7
-1	-1	-1	-1	0	1	1	1	1	q_1
1	1	$\boxed{0}$	-1	-1	-1	$\boxed{0}$	1	1	p

The two 0 entries in the bottom row prove the existence of two different zeros for $p(y,x)$ for each choice $y > 0$. □

We shall now state some immediate consequences of the previous theorem which provide different formulations of Tarski's Theorem. There is a geometric version in the spirit of semi-algebraic sets (cf. Definition **??**) as well as a logical version in the framework of quantifier elimination.

Theorem 1.2.13 (Tarski, geometric version, cf. [66] , [60]))
Let $A \subseteq \mathbb{R}^{n+1}$ be a semi-algebraic set; then both sets

$$\Pi A := \{y \in \mathbb{R}^n \mid \exists x \in \mathbb{R} \text{ such that } (y,x) \in A\} \quad \textit{"projection"}$$

and

$$\tilde{A} := \{y \in \mathbb{R}^n \mid \forall x \in \mathbb{R}\ (y,x) \in A\}$$

are semi-algebraic as well. Given a representation of A representations of ΠA and $\tilde{A}$ are computable by a BSS algorithm.

Proof. Without loss of generality let A be the finite intersection of sets of the form

$$\{(y,x) \in \mathbb{R}^{n+1} \mid \begin{array}{l} h_i(y,x) > 0\ ,\ i = 1,\ldots,s\ ;\ g_j(y,x) \geq 0\ ,\ j = 1,\ldots,l\ ; \\ f_k(y,x) = 0\ ,\ k = 1,\ldots,u\} \end{array}$$

(the projection of the union of such sets equals the union of the projections). Consider a polynomial family which includes precisely those polynomials h_i, g_j and f_k used in the description of at least one of the sets whose intersection constitute A. Define $\mathcal{W}$ to be the (finite) set of valid sign vectors for this family if evaluated in a point belonging to A. Now given a $w \in \mathcal{W}$ we can apply the algorithm of Theorem 1.2.10 to figure out all those sign patterns among $BC(\{h_i\}, 1 \leq i \leq s; \{g_j\}, 1 \leq j \leq l; \{f_k\}, 1 \leq k \leq u)$ which lead to a sign table $SGN(\{h_i\}, 1 \leq i \leq s; \{g_j\}, 1 \leq j \leq l; \{f_k\}, 1 \leq k \leq u)$ having a column with entry w. Let us denote this set of appropriate sign patterns by $\mathcal{E}_w$. Moreover, the possible choices c for variables y such that the above condition holds, build a semi-algebraic set. A description of it can be obtained from the algorithm as well: if we denote the polynomials in the base closure by $\{t_1, \ldots t_r\}$, then we obtain a description of the set of correct choices for c as

$$\{c \in \mathbb{R}^n \mid \bigvee_{\epsilon \in E_w} \bigwedge_{i=1}^{l} sgn(t_i(c)) = \epsilon_i\}\ .$$

Thus, the set is semi-algebraic.
$\tilde{A}$ is the complement of the semi-algebraic set $\{y \in \mathbb{R}^n \mid \exists x \ (y,x) \notin A\}$ and is therefore semi-algebraic as well.
Finally, both descriptions can be computed because the algorithm of Theorem 1.2.10 is a BSS algorithm. ■

Tarski's theorem is much more important than it might be realized when studying the above geometric formulation. We have already noticed that each semi-algebraic set can be written in the form

$$V \ = \ \{x \in \mathbb{R}^n \mid \psi(x)\} \ ,$$

where $\psi(x)$ is a Boolean combination of expressions of the form $p(x)\Delta 0$, $\Delta \in \{=, \geq, >\}$. If we split the variables into different blocks $\underline{x_1}, \ldots, \underline{x_k}, \underline{z}$ and then consider a *quantified first-order formula*

$$\exists \underline{x_1} \ \forall \underline{x_2} \ldots Q_k \underline{x_k} \ \psi(\underline{x_1}, \ldots, \underline{x_k}, \underline{z}) \quad ,$$

(where $Q_k \in \{\exists, \forall\}$), then Tarski's Theorem tells us that we can compute a description of the semi-algebraic (!) set

$$\{\underline{z} \mid \exists \underline{x_1} \ \forall \underline{x_2} \ldots Q_k \underline{x_k} \ \psi(\underline{x_1}, \ldots, \underline{x_k}, \underline{z}) \ \} \ .$$

Thus, the following theorem holds:

Theorem 1.2.14 (Tarski, model-theoretic version) The field of real numbers (or any other real closed field) admits *effective elimination of quantifiers* for its first-order logic. More precisely: given a first-order formula of the form

$$\rho(z_1, \ldots, z_n) \ \equiv \ Q_1 \ x_1 \ Q_2 \ x_2 \ldots Q_k \ x_k \ \psi(x_1, \ldots, x_k, z_1, \ldots, z_n) \ ,$$

where $Q_i \in \{\forall, \exists\}$, the z_i are free real variables and ψ is a first-order quantifier free Boolean formula with atomic predicates of the form

$$P_i(z_1, \ldots, z_n, x_1, \ldots, x_k) \ \Delta_i \ 0$$

with real polynomials $P_i, \Delta_i \in \{=, \geq, >\}$ for each (of finitely many) index i, we can compute (by a BSS algorithm) a first-order formula $\Theta(z)$ which is quantifier free and equivalent to ρ, i.e.

$$\forall \ z \in \mathbb{R}^n \ : \ \rho(z) \ \Leftrightarrow \ \Theta(z) \ .$$

■

The way we proved Tarski's theorem above following Muchnik's idea in addition gives, more or less directly, as well the similar result for *algebraically closed fields* .

Theorem 1.2.15 (Quantifier elimination in $\mathbb{C}$)
The field of complex numbers (or any other algebraically closed field) admits *effective elimination of quantifiers* for its first-order logic. More precisely: given a first-order formula of the form

$$\rho(z_1,\ldots,z_n) \;\equiv\; Q_1\; x_1\; Q_2\; x_2 \ldots Q_k\; x_k\; \psi(x_1,\ldots,x_k,z_1,\ldots,z_n)\;,$$

where $Q_i \in \{\forall,\exists\}$, the z_i are free (complex) variables and ψ is a first-order quantifier free Boolean formula with atomic predicates of the form

$$P_i(z_1,\ldots,z_n,x_1,\ldots,x_k)\;\Delta_i\;0$$

with complex polynomials $P_i, \Delta_i \in \{=,\neq\}$ for each (of finitely many) index i, we can compute (by a complex BSS algorithm) a first-order formula $\Theta(z)$ which is quantifier free and equivalent to ρ, i.e.

$$\forall\; z \in \mathbb{R}^n \;:\; \rho(z) \;\Leftrightarrow\; \Theta(z)\;.$$

■

Proof. The proof is almost similar to the real case. We just outline the differences. Given a finite family $\mathcal{P}$ of univariate complex polynomials $p_1,\ldots,p_s$ we define the *complex sign table* $\mathbb{C}\text{-}SGN(\mathcal{P})$ by changing Definition 1.2.1 as follows. If all the complex roots of the p_i are $z_1,\ldots,z_m$, then

$$\mathbb{C}\text{-}SGN(p_1,\ldots,p_s) := \left(\begin{pmatrix} sgn_{\mathbb{C}}\; p_1(z_1) \\ \vdots \\ sgn_{\mathbb{C}}\; p_s(z_1) \end{pmatrix}, \begin{pmatrix} sgn_{\mathbb{C}}\; p_1(z_2) \\ \vdots \\ sgn_{\mathbb{C}}\; p_s(z_2) \end{pmatrix}, \ldots, \begin{pmatrix} sgn_{\mathbb{C}}\; p_1(z_m) \\ \vdots \\ sgn_{\mathbb{C}}\; p_s(z_m) \end{pmatrix}\right),$$

Here, the *complex sign function* is defined as

$$sgn_{\mathbb{C}}(z) \;:=\; \begin{cases} 0 & z = 0 \\ 1 & z \neq 0 \end{cases}$$

In contrast to the real situation the ordering of the columns is irrelevant. We extend Definition 1.2.4 in a straightforward manner to obtain $\mathbb{C}\text{-}SGN(\mathcal{P},c)$ for families of multivariate polynomials and a $c \in \mathbb{C}^n$. The sets $C(\mathcal{P})$ and

$BC(\mathcal{P})$ are then defined as before. Our claim is that Theorem 1.2.10 holds as well for this adapted complex setting. Clearly, this is sufficient to prove the theorem.
In order to mimic the induction proof of Theorem 1.2.10 we have to describe how a complex sign table for $BC(\mathcal{P}) \cup \{q_1, \ldots, q_{i-1}\}$ can be extended to one for $BC(\mathcal{P}) \cup \{q_1, \ldots, q_i\}$. Actually, this step is easier than in the real setting. If $z_1, \ldots, z_m$ denote all the zeros of the polynomials $q_1, \ldots, q_{i+1}$, then we first compute the multiplicities μ_j of z_j as a zero of the new polynomial q_i. This can be done by inspecting the sign of $F_1(q_i)(z_j)$, of $F_1^2(q_i)(z_j)$ etc. until one of this derivatives is the first time not vanishing in z_j. Note that the polynomials $F_1(q_i), F_1^2(q_i), \ldots$ are already listet among $\{q_1, \ldots, q_{i-1}$. Since q_i has precisely $deg(q_i)$ many complex roots (counted with multiplicities) there remain

$$deg(q_i) - \sum_{j=1}^{m} \mu_j$$

many new roots to be added. All of them have multiplicity 1 because otherwise they would already appear among as root of $F_1(q_i)$ among $z_1, \ldots, z_m$. Therefore, we can add $deg(q_i) - \sum_{j=1}^{m} \mu_j$ many new columns to the complex sign table and compute the signs of $q_1, \ldots, q_{i-1}$ in the new zeros. ■

Let us come back to our initial question concerning decidability of all problems in class $\mathbf{NP}_{\mathbb{R}}$. The previous results imply:

Theorem 1.2.16 (Blum,Shub,Smale) All problems in class $\mathbf{NP}_{\mathbb{R}}$ are decidable. The same is true for problems in $\mathbf{NP}_{\mathbb{C}}$ in the complex BSS model.

Proof. Let (B, A) be a decision problem in $\mathbf{NP}_{\mathbb{R}}$ and let w be an input in B. According to Theorem **??** we can construct in polynomial time a polynomial $f(x)$ of degree at most four such that the initial question for w is equivalent to the problem whether the first-order formula

$$\exists x \in \mathbb{R}^n \; f(x) = 0$$

is true. The latter can be decided by Tarski's algorithm.
For the BSS model over $\mathbb{C}$, the problem of deciding the truth of a first-order formula

$$\exists \; x \in \mathbb{C}^n \;\; f_1(x) = 0 \;, \; \ldots \; , \; f_s(x) = 0$$

for degree two polynomials $f_i \in \mathbb{C}[x], 1 \leq i \leq s$ is $\mathbf{NP}_{\mathbb{C}}$-complete according to Remark **??**. This problem is decidable by a complex BSS machine by Theorem 1.2.15. ■

Tarski's theorem or, more general, questions about quantifier elimination are extremely important with respect to many questions concerning a complexity theory in general structures. We shall see a few more applications later on in this book.
One such application answers an earlier question concerning the relation between output sets and halting sets of BSS machines, see Definition **??**. The next result appeared in [8].

Proposition 1.2.17 The set of halting sets and the set of output sets of BSS machines over $\mathbb{R}^\infty$ are identical.

Proof. It is not hard to see that every halting set of a BSS machine M as well is an output set. Just build another machine $\tilde{M}$ which stores its input x in some predetermined registers during all its computation. The computation itself simulates the one of M on input x. If the latter stops, then $\tilde{M}$ writes the previously stored input x into its output registers and computes x as the result.
The reverse direction, however, is more difficult and can be done by applying Tarski's Theorem. Let M be a BSS machine with output set $\mathcal{O} := \phi_M(\mathbb{R}^\infty)$. We have to build a machine $\tilde{M}$ which on input $x \in \mathbb{R}^\infty$ decides whether $x \in \mathcal{O}$. If this is the case machine $\tilde{M}$ will halt its computation. Otherwise, it will continue forever in an endless loop. The problem we are faced to is that we do not know for which input y (or several inputs) machine M outputs x, or, more precisely, whether there is such an input at all. However, this can be decided using quantifier elimination.
Fix a natural number T. We first want to decide whether there is a computation of M outputting x and halting after at most T many steps. There is an effectively computable number $D(T) \in \mathbb{N}$ which denotes an upper bound on the dimension of those inputs for which M halts and outputs a result after at most M many steps. Therefore, x is the output of a computation of M in at most T many steps is and only if there exists an input $y \in \mathbb{R}^{D(T)}$ such that $\phi_M(y) = x$. Next, we consider a path γ which is a halting path after T many steps (this property is also decidable by a BSS program, given a description of the machine M). Using the path decomposition theorem we know that V_γ is semi-algebraic and that the function $\phi_M|_{V_\gamma}$ computed by M along γ is rational for every component of the result. If the latter

rational function for output component $i, 1 \leq i \leq D(T)$ is denoted by $\frac{f_{\gamma,i}}{g_{\gamma,i}}$, then there exists an input $y \in V_\gamma$ such that $\phi_M(y) = x$ if and only if

$$\exists\ y \in V_\gamma\ \wedge\ f_{\gamma,i}(y) = g_{\gamma,i}(y) \cdot x_i\ \ \forall\ 1 \leq i \leq size(x)\ .$$

Since V_γ is semi-algebraic and all $f_{\gamma,i}$ and $g_{\gamma,i}$ are polynomails, this question can be decided using Tarski's quantifier elimination procedure.
We can do the same for all halting path in T steps in order to decide whether x is outputted by a computation using T many steps. The question whether x is in $\mathcal{O}$ then can be decided by performing the above algorithm for $T = 1, T = 2, \ldots$. If $x \notin \mathcal{O}$ the algorithm will run forever, otherwise it will find a suitable value of T. ■

Some further results concerning the relation between halting and output sets over arbitrary structures are given in [48].

1.3 A simply exponential decision algorithm for problems in $\mathbf{NP}_{\mathbb{R}}$

Tarski's quantifier elimination procedure gives extremely bad complexity bounds; it's mainly of theoretical use. It is a stepwise elimination procedure in the sense that the variables are eliminated one after the other.

References

[1] Balcázar,J.L., Diaz,J., Gabarró,J.: *Structural Complexity I.* Springer (1988).

[2] Balcázar,J.L., Diaz,J., Gabarró,J.: *Structural Complexity II.* Springer (1990).

[3] Basu,S.: *New Results on Quantifier Elimination over Real Closed Fields and Applications to Constraint Databases.* Journal of the ACM, Vol. 46, No. 4, 537–555 (1999).

[4] Basu,S., Pollack,R., Roy,M.F.: *On the combinatorial and algebraic complexity of quantifier elimination.* Journal of the ACM, 43(6), 1002–1045 (1996).

[5] Benedetti,R., Risler,J.J.: *Real algebraic and semi-algebraic sets.* Hermann, Paris (1990).

[6] Ben-Or,M., Kozen,D., Reif,J.: *The Complexity of Elementary Algebra and Geometry.* Journal of Computer and System Sciences, 32, 251–264 (1986).

[7] Blum,L., Cucker,F., Shub,M., Smale,S.: *Complexity and real computation.* Springer (1997).

[8] Blum,L., Shub,M., Smale,S.: *On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines.* Bull. Amer. Math. Soc., 21, 1–46 (1989).

[9] Bochnak,J., Coste,M., Roy,M.-F.: *Géometrie algébrique réelle.* Springer, Berlin (1987).

[10] Borgwardt,K.H.: *The Average Number of Pivot Steps Required by the Simplex-Method is Polynomial.* Zeitschrift f"ur Operations Research, Vol. 7, No. 3, 157–177 (1982).

[11] Borgwardt,K.H.: *The Simplex Method.* Springer (1987).

[12] Bürgisser,P., Clausen,M., Shokrollahi,A.: *Algebraic Complexity Theory.*, Grundlehren der mathematischen Wissenschaften, vol. 315, Springer (1996).

[13] Canny,J.: *Generalized characteristic polynomials.* Journal of Symbolic Computations, 9, 241–250 (1990).

[14] Collins,G.E.: *Quantifier Elimination for real closed fields by Cylindrical Algebraic Decomposition.* Springer Lecture Notes in Computer Science vol. 33, 134–183 (1977).

[15] Cook,S.: *The Complexity of Theorem-Proving Procedures.* Proc. of the 3^{rd} Annual ACM Symposium on Theory of Computing STOC, ACM, 171 – 158 (1971).

[16] Cucker, F.: *On the complexity of quantifier elimination : The structural approach.* The Computer Journal, vol. 36 No. 5, 400–408 (1993)

[17] Coste,M., Roy,M.F.: *Thom's Lemma, the coding of Real Algebraic Numbers.* Journal of Symbolic computation,5, 121–129 (1988).

[18] Cucker, F., Rossello, F.: *On the complexity of some problems for the Blum-Shub-Smale model.* Proc. LATIN'92, Lecture Notes in Computer Science vol583, 117–129 (1992).

[19] Cucker,F., Shub,M., Smale,S.: *Separation of complexity classes in Koiran's weak model.* Theoretical Computer Science, 133, 3–14 (1994).

[20] Cucker,F., Smale,S.: *Complexity Estimates Depending on Condition and Round-off Error.* Journal of the ACM, 46 (1), 113–184 (1999).

[21] Davenport, J, Heintz,J.: *Real quantifier elimination is doubly exponential.* Journal of Symbolic Computation 5, 29–35 (1988).

[22] Eaves,B.C.: *On Quadratic Programming.* Management Science (Theory) 17, No. 11, 698–711 (1971).

[23] Even,S., Itai,A., Shamir,A.: *On the complexity of timetable and multicommodity flow problems.* SIAM Journal on Computing , 691–703 (1976).

[24] Fournier,H., Koiran,P.: *Are lower bounds easier over the reals?* Proc. of the 30^{th} Annual ACM Symposium on Theory of Computing, 507–513 (1998).

[25] Fournier,H., Koiran,P.: *Lower Bounds Are not easier over the reals: Inside PH.* Proc. ICALP, XX–XX (2000).

[26] Garey,M.R., Johnson,D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco (1979).

[27] Grädel,E., Meer,K.: *Descriptive Complexity Theory over the Real Numbers*, in: Proceedings of the AMS Summer Seminar on "Mathematics of Numerical Analysis: Real Number Algorithms", Park City 1995, Lectures in Applied Mathematics, eds.: J. Renegar, M. Shub, S. Smale, 381–404 (1996).

[28] Grigor'ev,D.Y.: *Complexity of Deciding Tarski Algebra.* Journal of Symbolic Computation, 5, 65–108 (1988).

[29] Heintz,J., Roy,M.F., Solerno,P.: *On the complexity of semialgebraic sets.* Proceedings IFIP 1989, San Fracisco, North-Holland 293–298 (1989).

[30] Hemmerling,A.: *Computability and Complexity over Structures of Finite Type.* E.M.Arndt-University Greifswald, Preprint nr. 2 (1995).

[31] Hörmander,L.: *The analysis of linear partial differential operators. Vol. 2* Springer (1983).

[32] Immerman,N.: *Descriptive Complexity.* Springer Graduate Texts in Computer Science (1999).

[33] Karmarkar,N.: *A new polynomial-time algorithm for linear programming.* Combinatorica 4(4), 373–385 (1984).

[34] Karp,R.M.: *Reducibility among combinatorial problems.* In: Miller,R.E., Thatcher,J.W. (eds.), Complexity of Computer Computations, Plenum Press, New York, 85–103 (1972).

[35] Khachiyan,L.G.: *A polynomial algorithm in linear programming.* Dokl. Akad. Nauk, 244, 1093–1096 (1979).

[36] Koiran,P.: *A weak version of the Blum-Shub-Smale model.* Proceedings of the ACM Conference on Foundations of Computer Science FOCS'93, 486–495 (1993).

[37] Koiran,P.: *Eliminations of Constants from Machines over Algebraically Closed Fields.* Journal of Complexity 13, 65–82 (1997)

[38] Lickteig,T.: *On semialgebraic decision complexity.* TR-90-052 ICSI, Berkeley und Univesität Tübingen, Habilitationsschrift (1990).

[39] Lickteig,T.: *Semi-algebraic Decision Complexity, the Real Spectrum and Degree*, Journal of pure and applied algebra 110, 2, 131–184 (1996).

[40] Łojasiewicz,S.: *Triangulation of semi-algebraic sets.* Ann. Scuola Norm. Sup. Pisa, 3, 18, 449–474 (1964).

[41] Macauley,F.: *The algebraic theory of modular systems.* Cambridge University Press, Cambridge UK (1916).

[42] Matijasevich,Y.V.: *Enumerable Sets are Diophantine.* Dokl. Akad. Nauk SSSR, 191, 279–282 (1970).

[43] Meer,K.: *Computations over $\mathbb{Z}$ and $\mathbb{R}$: a comparison.* Journal of Complexity, 6, 256–263 (1990).

[44] Meer,K.: *On the complexity of Quadratic Programming in real number models of computations.* Theoretical Computer Science, 133, 85–94 (1994).

[45] Meer,K., Michaux,C.: *A Survey on Real Structural Complexity Theory.* Bulletin of the Belgian Mathematical Society - Simon Stevin, 4, 113–148 (1997).

[46] Megiddo,N.: *A general NP-completeness Theorem.* in : From Topology to Computation, Proceedings of the Smalefest, 432–442, Springer (1993).

[47] Michaux,C.: *Une remarque à propos des machines sur $\mathbb{R}$ introduites par Blum, Shub et Smale.* C.R. Acad. Sci. Paris, t. 309, série I, 435–437 (1989).

[48] Michaux,C.: *Ordered rings over which output sets are recursively enumerable.* Proceedings of the AMS 111, 569–575 (1991).

[49] Michaux,C.: *$P \neq NP$ over the nonstandard reals implies $P \neq NP$ over $\mathbb{R}$.* Theoretical Computer Science, 133, 95–104 (1994).

[50] Michaux,C., Öztürk,A.: *Quantifier Elimination following Muchnik.* Université Mons-Hainaut, Preprint (1999).

[51] Novak,E.: *The real number model in numerical analysis.* Journal of Complexity, 11, 57–73 (1995).

[52] Papadimitriou,C.H.: *Computational Complexity.* Addison-Wesley (1994).

[53] Paul.W.: *Komplexitätstheorie.* Teubner Verlag, Stuttgart (1978).

[54] Poizat,B.: *Les Petits Cailloux, une approche modèle-théorique de l'Algorithmie.* Aléas (1995).

[55] Renegar,J.: *A polynomial-time algorithm, based on Newton's method, for linear programming.* Mathematical Programming 40, 59 – 93 (1988)

[56] Renegar,J.: *Recent progress on the complexity of the decision problem for the reals.* DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 6, 287–308 (1991).

[57] Renegar,J.: *On the computational Complexity and Geometry of the first-order Theory of the Reals , I - III.* Journal of Symbolic Computation, 13, 255–352 (1992).

[58] Schöning,U.: *A Low and a High Hierarchy in NP.* Journal of Computer and System Sciences, 27, 14–28 (1983).

[59] Schrijver,A.: *Theory of Linear and Integer Programming.* Wiley, New-York (1986).

[60] Seidenberg,A.: *A new decision method for elementary algebra.* Annals of Mathematics vol60 no. 2, 365 - 374 (1954).

[61] Shub,M., Smale,S.: *Complexity of Bezout's Theorem I : Geometric aspects.* Journal of the AMS, 6, 459–501 (1993).

[62] Smale,S.: *On the average number of steps of the simplex method of linear programming.* Mathematical Programming 27, 241–262 (1983).

[63] Smale,S.: *Complexity Theory and Numerical Analysis.* Acta Numerica, 6, 523–551 (1997).

[64] Sturm,C.: *Mémoire sur la résolution des équations numerique,* Inst. France Sci. Math. Phys. 6 (1835).

[65] Tardos,E.: *A strongly polynomial algorithm to solve combinatorial linear programs.* Operations Research, 34(2), 250–256 (1986).

[66] Tarski,A.: *A decision method for elementary algebra and geometry.*, 2nd edition, Univ. Calif. Press, Berkeley (1951).

[67] Traub,J.F., Wasilkowski,G.W., Woźniakowski,H.: *Information-based complexity.* Academic Press (1988).

[68] Traub,J.F., Werschulz,A.G.: *Complexity and Information.* Cambridge University Press (1998).

[69] Traub,J.F., Woźniakowski,H.: *Complexity of linear programming.* Operations Research Letters, 1(2), 59–62 (1982).

[70] Triesch,E.: *A note on a Theorem of Blum, Shub, and Smale.* Journal of Complexity, 6, 166-169 (1990).

[71] Turing,A.: *On computable numbers, with an application to the Entscheidungsproblem.* Proc. London Mathematical Society, Series 2 42, 230-265 (1936).

[72] Vavasis,S.A.: *Nonlinear Optimization, Complexity Issues.* Oxford University Press (1991).

[73] Vavasis,S.A., Ye,Y.: *A primal-dual interior-point method whose running time depends only on the constarint matrix.* Mathematical Programming, 74, 79–120 (1996).

[74] Weispfenning.V.: *The complexity of linear problems in fields.* Journal of Symbolic Computation 5, 3–27 (1988).

[75] Woźniakowski,H.: *Why does information-based complexity use the real number model?* Theoretical Computer Science, 219 (1-2), 451–465 (1999).

[76] Wüthrich,H.R.: *Ein Entscheidungsverfahren für die Theorie der reell-abgeschlossenen Körper.* Springer Lecture Notes in Computer Science vol. 43, 138–162 (1976).