Fachbereich Informatik der Universität Hamburg

Vogt-Kölln-Str. 30 ◊ D-22527 Hamburg / Germany

University of Hamburg - Computer Science Department

Mitteilung Nr. 333/05 • Memo No. 333/05

Quantifier Elimination over Real Closed Fields in the Context of Applied Description Logics

Claudia Schultz, Ralf Möller

Arbeitsbereich KOGS

FBI-HH-M-333/05

September 2005

Abstract

This work investigates the quantifier elimination problem in real closed fields with respect to the application in description logics. The motivation for the investigation in this topic is mainly based on the demand for an extension of the description logic system RACER to support a default concrete domain for non-linear multivariate equations and inequations. In this report we provide a literature overview that summarizes main mathematical tools for checking whether a set of non-linear multivariate (in-)equations is satisfiable (quantifier elimination problem). In addition, the report describes the interface to a prototype implementation for complex numbers (rather than the reals) provided by RACER. We also shortly describe how initial application examples are handled with the prototype implementation.

Zusammenfassung

In dieser Arbeit wird das Quantoreneliminationsproblem in rellen abgeschlossenen Körpern im Zusammenhang mit Beschreibungslogiken untersucht. Die Untersuchung dieses Themas ist im wesentlichen durch die Forderung motiviert, das Beschreibungslogiksystem RACER um eine Komponente zu erweitern, die konkrete Domänen mit linearen multivariaten Gleichungen und Ungleichungen standardmäßig unterstützt. Dieser Beitrag bietet einen Literaturüberblick über die wichtigsten mathematischen Werkzeuge zur Überprüfung der Erfüllbarkeit von nichtlinearen multivariaten (Un-)gleichungen (Quantoreneliminationsproblem). In dem Report wird außerdem die Schnittstelle der Prototypimplementation für den Körper der komplexen Zahlen beschrieben. Weiterhin werden erste Anwendungsbeispiele erläutert.

Chapter 1

Introduction: The Decision Problem in the Context of Description Logics

This work investigates the quantifier elimination problem in real closed fields with respect to the application in description logics. The motivation for the investigation in this topic is mainly based on the demand for an extension of the description logic system RACER[1] (Renamed ABox and Concept Expression Reasoner) to support a default concrete domain for non-linear multivariate equations and inequations. It is essential for practical applications to be able to reason about objects from other domains, so called concrete domains (see, e.g., [HMW01]). RACER provides TBox and ABox reasoning for the very expressive description logic $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$. We assume the reader has basic knowledge about description logic. As a summary a brief introduction to the description logic $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ is given, followed by an illustrating example for the use of a concrete domain for non-linear multivariate equations and inequations.

In description logic (DL) systems, a knowledge base consists of a TBox and an ABox. A TBox specifies the set of terminological axioms which represents the conceptual knowledge of an application domain, while an ABox is a set of assertional axioms denoting the knowledge about the instances of the domain and their interrelationships. Each DL system includes the three major logic relations: subsumption (inclusion), equivalence, and disjointness. For domain modeling, $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ provides five disjoint sets: a set of concept names C, a set of role names R, a set of feature names F, a set of individual names O and a set of names for (concrete) objects O_C. Figure 1.1 illustrates the syntax and the semantic of the DL $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$.

The set of concept names C represents atomic concepts, and the set of role names R indicates atomic roles. Roles are binary predicates that describe relations between elements of O. The right hand side of a role is called a role filler . The set of role names R can be divided into a set of non-transitive roles P and a set of transitive roles T such that $R = P \cup T$ and P and T are disjoint. Features, also called attributes, are functional roles, i.e. each individual can only have up to one filler for this role. The set of features F is a subset of P. Elements of $F \cup R$ are atomic roles .

[1] RACER download page: `http://www.racer-systems.com/`

(a)

Syntax	Semantics
Concepts ($\mathsf{R} \in R$, $\mathsf{S} \in S$, and $\mathsf{f}, \mathsf{f_i} \in F$)	
A	$\mathsf{A}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ (A *is a concept name*)
$\neg\mathsf{C}$	$\Delta^{\mathcal{I}} \setminus \mathsf{C}^{\mathcal{I}}$
$\mathsf{C} \sqcap \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cap \mathsf{D}^{\mathcal{I}}$
$\mathsf{C} \sqcup \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \cup \mathsf{D}^{\mathcal{I}}$
$\exists\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists\,\mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \wedge \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\forall\,\mathsf{R}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall\,\mathsf{b} \in \Delta^{\mathcal{I}} : (\mathsf{a},\mathsf{b}) \in \mathsf{R}^{\mathcal{I}} \Rightarrow \mathsf{b} \in \mathsf{C}^{\mathcal{I}}\}$
$\exists_{\geq n}\,\mathsf{S}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \|\{\mathsf{y} \mid (\mathsf{x},\mathsf{y}) \in \mathsf{S}^{\mathcal{I}}, \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\| \geq \mathsf{n}\}$
$\exists_{\leq m}\,\mathsf{S}\,.\,\mathsf{C}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \|\{\mathsf{y} \mid (\mathsf{x},\mathsf{y}) \in \mathsf{S}^{\mathcal{I}}, \mathsf{y} \in \mathsf{C}^{\mathcal{I}}\}\| \leq \mathsf{m}\}$
$\exists\,\mathsf{f_1},\ldots,\mathsf{f_n}\,.\,\mathsf{P}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \exists\,\mathsf{x_1},\ldots,\mathsf{x_n} \in \Delta^{\mathcal{D}} : (\mathsf{a},\mathsf{x_1}) \in \mathsf{f_1}^{\mathcal{I}} \wedge \ldots \wedge (\mathsf{a},\mathsf{x_n}) \in \mathsf{f_n}^{\mathcal{I}} \wedge (\mathsf{x_1},\ldots,\mathsf{x_n}) \in \mathsf{P}^{\mathcal{I}}\}$
$\forall\,\mathsf{f_1},\ldots,\mathsf{f_n}\,.\,\mathsf{P}$	$\{\mathsf{a} \in \Delta^{\mathcal{I}} \mid \forall\,\mathsf{x_1},\ldots,\mathsf{x_n} \in \Delta^{\mathcal{D}} : (\mathsf{a},\mathsf{x_1}) \in \mathsf{f_1}^{\mathcal{I}} \wedge \ldots \wedge (\mathsf{a},\mathsf{x_n}) \in \mathsf{f_n}^{\mathcal{I}} \Rightarrow (\mathsf{x_1},\ldots,\mathsf{x_n}) \in \mathsf{P}^{\mathcal{I}}\}$
Roles and Features	
R	$\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
f	$\mathsf{f}^{\mathcal{I}} : \Delta^{\mathcal{I}} \rightarrow \Delta^{\mathcal{D}}$ (features are partial functions)

$\|\cdot\|$ denotes the cardinality of a set, and $n, m \in \mathbb{N}$ with $n > 1$, $m > 0$.

(b)

Axioms	
Syntax	Satisfied if
$\mathsf{R} \in T$	$\mathsf{R}^{\mathcal{I}} = (\mathsf{R}^{\mathcal{I}})^{+}$
$\mathsf{R} \sqsubseteq \mathsf{S}$	$\mathsf{R}^{\mathcal{I}} \subseteq \mathsf{S}^{\mathcal{I}}$
$\mathsf{C} \sqsubseteq \mathsf{D}$	$\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{D}^{\mathcal{I}}$

(c)

Assertions ($\mathsf{a},\mathsf{b} \in O$, $\mathsf{x},\mathsf{x_i} \in O_C$)	
Syntax	Satisfied if
$\mathsf{a}:\mathsf{C}$	$\mathsf{a}^{\mathcal{I}} \in \mathsf{C}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{b}):\mathsf{R}$	$(\mathsf{a}^{\mathcal{I}},\mathsf{b}^{\mathcal{I}}) \in \mathsf{R}^{\mathcal{I}}$
$(\mathsf{a},\mathsf{x}):\mathsf{f}$	$(\mathsf{a}^{\mathcal{I}},\alpha(\mathsf{x})) \in \mathsf{f}^{\mathcal{I}}$
$(\mathsf{x_1},\ldots,\mathsf{x_n}):\mathsf{P}$	$(\alpha(\mathsf{x_1}),\ldots,\alpha(\mathsf{x_n})) \in \mathsf{P}^{\mathcal{I}}$

Figure 1.1: Syntax and Semantics of $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ (Figure taken from [HM02]).

Concepts can be built from atomic concepts using boolean operators including negation, conjunction and disjunction. In addition, concepts can be constructed using existential and value restrictions, qualified number restrictions, and predicate restrictions. Value restrictions are applied to enforce that all role fillers of a certain role are of a specific concept. Let $\mathsf{R}, \mathsf{S} \in R$ be role names and let $\mathsf{C} \in C$ be a concept name in the following. Then, the DL notation for this is $\forall\,\mathsf{R}\,.\,\mathsf{C}$. For exists restrictions $\exists\,\mathsf{R}\,.\,\mathsf{C}$, it is required that there exists at least one role filler for a certain role which is of the stated concept. Qualified number restrictions ensure an upper or lower bound for the number of role fillers of a certain role that each instance of the indicated concept must have. They are represented by $\exists_{\geq n}\,\mathsf{S}\,.\,\mathsf{C}$ and $\exists_{\leq m}\,\mathsf{S}\,.\,\mathsf{C}$, respectively. For features of a concrete domain, predicate exists and predicate all restrictions are provided, denoted by $\exists\,\mathsf{f_1},\ldots,\mathsf{f_n}\,.\,\mathsf{P}$ and $\forall\,\mathsf{f_1},\ldots,\mathsf{f_n}\,.\,\mathsf{P}$, respectively. The concepts are presented in Figure 1.1a.

Roles can only be built from atomic roles with the inverse role constructor. The inverse of a role R is then denoted by R^{-1}.

A role hierarchy $\mathcal{R}$ can be generated via role inclusion axioms, for example $\mathsf{S} \sqsubseteq \mathsf{R}$. The role S is then called a sub-role of R and R is a super-role of S. Define $\sqsubseteq^*$ as the reflexive transitive closure of $\sqsubseteq$ over $\mathcal{R}$, the set of sub-roles $\mathsf{R}^{\downarrow}$ of a role R is given by $\mathsf{R}^{\downarrow} = \{\mathsf{S} \in R \mid \mathsf{S} \sqsubseteq^* \mathsf{R}\}$. The set of super-roles can be specified accordingly. Simple roles are roles which are neither transitive nor have a transitive role as a sub-role. Thus, the set of simple roles is $S := \{\mathsf{R} \in P \mid \mathsf{R}^{\downarrow} \cap T = \emptyset\}$. Qualified number restrictions can only be expressed for simple roles.

A TBox is then a set of general concept inclusions (GCI's), i.e. concept inclusion axioms, role inclusion axioms, and role element axioms (see Figure 1.1b). They are denoted by $\mathsf{C} \sqsubseteq \mathsf{D}$, $\mathsf{R} \sqsubseteq \mathsf{S}$, and $\mathsf{R} \in T$, respectively. The equivalence relation, denoted by $\mathsf{C} \equiv \mathsf{D}$, is an abbreviation for $\mathsf{C} \sqsubseteq \mathsf{D}$ and $\mathsf{D} \sqsubseteq \mathsf{C}$, whereas the disjointness relation can be expressed by $\mathsf{C} \sqsubseteq \neg\, \mathsf{D}$. In addition, the concepts "top" and "bottom" are abbriviated by ($\top$) and ($\bot$). The former represents the top-most concept in the hierarchy, while the latter stands for the inconsistent concept. In a TBox, it is also possible to define domain and range restrictions for roles in order to restrict the left and right side of an atomic role. This can be implemented using GCIs, i.e. $(\exists\, \mathsf{R}\,.\,\top) \sqsubseteq \mathsf{C}$ for domain restrictions and $\top \sqsubseteq (\forall\, \mathsf{R}\,.\,\mathsf{D})$ for range restrictions.

The semantics of $\mathcal{ALCQHI}_{R^+}(\mathcal{D})^-$ is given as a Tarski-style model theoretic semantics. $\mathcal{I}_\mathcal{D} = (\Delta^\mathcal{I}, \Delta_\mathcal{D}, \cdot^\mathcal{I})$ is an interpretation that consists of a set $\Delta^\mathcal{I}$, the abstract domain, a set $\Delta_\mathcal{D}$, the domain of the concrete domain $\mathcal{D}$ and an interpretation function $\cdot^\mathcal{I}$, where $\Delta^\mathcal{I} \cap \Delta_\mathcal{D} = \emptyset$ holds. The interpretation function maps each concept name C to a subset $\mathsf{C}^\mathcal{I}$ of $\Delta^\mathcal{I}$, each role name R to a subset $\mathsf{R}^\mathcal{I}$ of $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ and each feature f from F to a partial function $\mathsf{f}^\mathcal{I}$ from $\Delta^\mathcal{I} \to \Delta_\mathcal{D}$. A concrete domain $\mathcal{D}$ is then defined as a pair $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$, where $\Phi_\mathcal{D}$ is a set of predicate names and $\Delta_\mathcal{D}$ denotes a specific concrete domain. The interpretation function maps each predicate name P from $\Phi_\mathcal{D}$ with arity n to a subset $\mathsf{P}^\mathcal{I}$ of $\Delta^n_\mathcal{D}$. The predicate $\top_\mathcal{D}$ specifies the predicate which is true for all elements in $\Delta_\mathcal{D}$. Assume that $\bot_\mathcal{D}$ is the negation of the predicate $\top_\mathcal{D}$. A concrete domain $\mathcal{D}$ is called admissible iff the set of predicate names $\Phi_\mathcal{D}$ is closed under negation and $\Phi_\mathcal{D}$ contains a name $\top_\mathcal{D}$ for $\Delta_\mathcal{D}$, and the satisfiability problem $\mathsf{P}_1^{\mathsf{n}_1}(\mathsf{x}_{11}, \ldots, \mathsf{x}_{1\mathsf{n}_1}) \wedge \ldots \wedge \mathsf{P}_\mathsf{m}^{\mathsf{n}_\mathsf{m}}(\mathsf{x}_{\mathsf{m}1}, \ldots, \mathsf{x}_{\mathsf{m}\mathsf{n}_\mathsf{m}})$ is decidable (m is finite, $\mathsf{P}_\mathsf{i}^{\mathsf{n}_\mathsf{i}} \in \Phi_\mathcal{D}$, n_i is the arity of P_i, and x_jk is a concrete object). Therefore, a variable assignment α maps concrete objects to values in $\Delta_\mathcal{D}$.

An ABox is a finite set of assertional axioms (see Figure 1.1c). Let C be a concept term, R be a role, $\mathsf{a}, \mathsf{b} \in O$ be individual names and $\mathsf{x}, \mathsf{x}_1, \cdots, \mathsf{x}_\mathsf{n} \in O_C$ be names for concrete objects. Then $\mathsf{a}:\mathsf{C}$ indicates a concept assertion, $(\mathsf{a}, \mathsf{b}):\mathsf{R}$ denotes a role assertion, $(\mathsf{a}, \mathsf{x}):\mathsf{f}$ represents a concrete domain feature assertion and $(\mathsf{x}_1, \ldots, \mathsf{x}_\mathsf{n}):\mathsf{P}$ defines a concrete domain predicate assertion. The interpretation function $\cdot^\mathcal{I}$ of the interpretation $\mathcal{I}_\mathcal{D}$ can now be extended by mapping each individual name from O to an element of $\Delta^\mathcal{I}$ and each name for concrete objects from O_C to an element of $\Delta^\mathcal{D}$. Note that different individuals are mapped to different domain objects (unique name assumption).

An interpretation $\mathcal{I}$ is a model of a concept C (or satisfies a concept C) iff $\mathsf{C}^\mathcal{I} \neq \emptyset$ and for all $\mathsf{R} \in R$ it holds that iff $(x, y) \in \mathsf{R}^\mathcal{I}$ then $(y, x) \in (\mathsf{R}^{-1})^\mathcal{I}$. An interpretation $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ iff it satisfies all axioms in $\mathcal{T}$ (see Figure 1.1b). An interpretation $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ w.r.t. a TBox $\mathcal{T}$ iff it is a model of $\mathcal{T}$ and satisfies all assertions in $\mathcal{A}$ (see Figure 1.1c).

The ABox consistency problem is to decide whether a given ABox $\mathcal{A}$ is consistent with respect to a TBox $\mathcal{T}$. An ABox is consistent with respect to a TBox $\mathcal{T}$ if and only if it has a model with respect to $\mathcal{T}$. Otherwise, the ABox is called inconsistent.

An example is presented for the use of the concrete domain $(\Delta_\mathcal{D}, \Phi_\mathcal{D})$ with $\Delta_\mathcal{D} = \mathbb{R}$ and a set of predicates $\Phi_\mathcal{D}$ for non-linear equations and inequations between real numbers. For sake of readability and brevity, predicates are denoted as lambda expressions instead of introducing predicate names. Let $\mathsf{x}_1\mathsf{pos\_circle}$, $\mathsf{x}_2\mathsf{pos\_circle}$, $\mathsf{x}_1\mathsf{pos\_hyperbola}$ and

$\mathsf{x}_2\mathsf{pos\_hyperbola}$ be features. Suppose the following GCIs are in a TBox:

$\textbf{human\_in\_circle} \equiv \mathsf{human} \sqcap \exists\, \mathsf{x}_1\mathsf{pos\_circle}, \mathsf{x}_2\mathsf{pos\_circle} \,.\, \lambda(x_1, x_2)(x_1^2 + x_2^2 - 1 < 0)$
$\textbf{human\_in\_on\_circle} \equiv \mathsf{human} \sqcap \exists\, \mathsf{x}_1\mathsf{pos\_circle}, \mathsf{x}_2\mathsf{pos\_circle} \,.\, \lambda(x_1, x_2)(x_1^2 + x_2^2 - 1 \leq 0)$
$\textbf{human\_on\_hyperbola} \equiv \mathsf{human} \sqcap \exists\, \mathsf{x}_1\mathsf{pos\_hyperbola}, \mathsf{x}_2\mathsf{pos\_hyperbola} \,.\, \lambda(x_1, x_2)(x_1^3 - x_2^2 = 0)$

A human in a circle is a human who is in a geometric circle but not on the circular line itself, while a human in or on a circle is a human who can be in a circle or on the circular line. A human on a hyperbola is a human who is on the branches of a geometric hyperbola. Obviously, the concept $\mathsf{human\_in\_circle}$ is subsumed by the concept $\mathsf{human\_in\_on\_circle}$. With the TBox defined above, subsequent constraints between the individuals lara and leon are established in the ABox.

$\mathsf{lara} : \mathsf{human\_in\_circle}$
$(\mathsf{lara}, \mathsf{x}_1\mathsf{pos\_lara}) : \mathsf{x}_1\mathsf{pos\_circle}, (\mathsf{lara}, \mathsf{x}_2\mathsf{pos\_lara}) : \mathsf{x}_2\mathsf{pos\_circle}$
$\mathsf{leon} : \mathsf{human\_on\_hyperbola}$
$(\mathsf{leon}, \mathsf{x}_1\mathsf{pos\_leon}) : \mathsf{x}_1\mathsf{pos\_hyperbola}, (\mathsf{leon}, \mathsf{x}_2\mathsf{pos\_leon}) : \mathsf{x}_2\mathsf{pos\_hyperbola}$
$(\mathsf{x}_1\mathsf{pos\_lara}, \mathsf{x}_1\mathsf{pos\_leon}) : \lambda(x_1, x_2)(x_1 = x_2)$
$(\mathsf{x}_2\mathsf{pos\_lara}, \mathsf{x}_2\mathsf{pos\_leon}) : \lambda(x_1, x_2)(x_1 = x_2)$

To decide the ABox consistency problem, one has to deduce that the ABox is consistent with respect to the TBox defined above. This implies that lara and leon can be on the same position. To deduce the consistency of the ABox means to solve the non-linear system of polynomial equations and inequations containing the defining polynomial inequation for the circle and the defining polynomial equation for the hyperbola:

$$x_1^2 + x_2^2 - 1 < 0 \tag{1.1}$$
$$x_1^3 - x_2^2 = 0. \tag{1.2}$$

Figure 1.2 illustrates these defining polynomial equations and inequations.

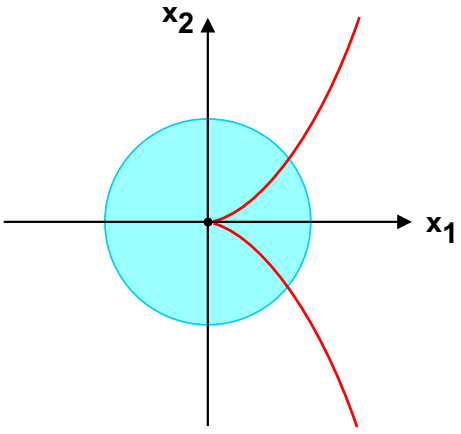

Figure 1.2: Illustration of the polynomial system.

Solutions of Inequation (1.1) are all points which lie in the shadowed circle without the circle itself. All valid solutions of Equation (1.2) are depicted in the two hyperbola branches.

Obviously, the common solutions are represented by the parts of the hypberbola branches which lie in, but not on, the circle. Thus, the ABox $\mathcal{A}$ should be verified to be consistent.

The main task of this work is to find and analyze computationally efficient algorithms which can answer the question whether a given polynomial system is consistent with respect to solutions in real closed fields. For example, the set of real numbers is a real closed field. An exact definition, also for polynomial systems, will be given in the mathematical overview in Chapter 2. This question is often referred to as the decision problem. Basically, there exist three different interpretations for the question whether a system of polynomial equations and inequations can be solved:

1) **Decision interpretation**, also called **existential interpretation**: This interpretation asks for the consistency of a given polynomial system, which means just to answer the question whether a polynomial system has a common solution or not.

2) **Counting interpretation**: The counting interpretation ask for the number of solutions, i.e. the cardinality of the solution set, in case the dimension of the solution set is zero. A zero-dimensional solution set indicates that the number of solutions is finite. If the cardinality of the solution set is infinite, the task is refined to the task of determining the dimension of the solution set.

3) **Enumeration interpretation**: This interpretation requires to compute the exact solutions in case the solution set is finite. If the solution set is infinite, the task is to find sample points or to describe each connected component of the solution set. For example, if the intersection of two planes results in a line, the line is a connected component of the solution set which has to be described.

The decision problem can be seen as a quantifier elimination (QE) problem. The quantifier elimination problem for the real numbers poses the task to find a quantifier-free formula ϕ for a given formula φ such that ϕ is always true if and only if φ is true in the domain of real numbers. Quantifier elimination is based on first-order logic combined with polynomial equations and inequations. Atomic formulas are defined as polynomial equations and inequations of the form $f > 0$, $f \geq 0$, $f = 0$, $f < 0$, $f \leq 0$ and $f \neq 0$. (Note that it would also be sufficient to allow only the two forms $f > 0$ and $f = 0$ since the other forms can be expressed by those two forms.) A quantifier-free formula is then a boolean combination of atomic formulas, where the boolean operators can be elements of $\{\wedge, \vee, \neg\}$. A quantifier-free formula can be quantified by universal $\forall x$ and/or existential quantifiers $\exists x$. Such a formula is therefore called a quantified formula. All variables which are not quantified in a formula are named parameters or free variables. Quantified formulas which contain only existential or only universal quantifiers are called existential or universal formulas, respectively. $\exists x_1 \cdots \exists x_n \phi(u_1, \cdots, u_m, x_1, \cdots, x_n)$ is an example for a existential formula, where $u_1, \cdots, u_m$ denote the parameters and $x_1, \cdots, x_n$ indicates the variables of a quantifier-free formula ϕ. In general, a first-order formula is constructed by alternating blocks of universal and existential quantifiers in front of a quantifier-free formula. For example, applying quantifier elimination to the first-order formula $a \neq 0 \wedge \exists x(ax^2+bx+c = 0)$ yield the equivalent quantifier-free formula $b^2 - 4ac \geq 0$, which is true if and only if the quantified formula has a real solution. The quantifier elimination problem for the real numbers can always be solved as first stated and proved by Tarski in 1951 [Tar51]. However, Tarski's QE algorithm was not constructive. The decision problem as considered

in the application of description logics is only concerned with existential formulas. Thus quantifier elimination does not really apply.

Algebra is that part of mathematics that is originally concerned with equations and their solutions. The name Algebra was first used in the 12th century as a translation of the Arabic word "al-dschabr" which occured in the title of a book from the 9th century explaining calculations with equations. Calculation involving letters (variables) was then established in the 16th century by Francois Viète (1540-1603). Modern algebra investigates structures of mathematical sets whereas linear algebra was concerned with solving problems in geometry and therefore with solving systems of linear equations. Nowadays, computer algebra, also called algorithmic algebra, is one of the most interesting fields between mathematics and computer science. An often cited definition for computer algebra is the following: "Computer algebra is that part of computer science which designs, analyzes, implements, and applies algebraic algorithms" [BCL82]. Computer algebra puts an emphasis on symbolic and algebraic computations, but one can also find computer algebra algorithms dealing with integral, differential, and numerical computations. Thus, the term computer algebra includes a wide range of mathematical fields. However, all algorithms in computer algebra have a preceeding algebraic structure analysis in common.

The history of solving systems of multivariate polynomial equations dates back to the 19th century. The classical elimination theory is based on resultants. A definition of resultants can be cited from Petitjean [Pet97]: "A resultant is an algebraic criterion for determining when a pair of univariate polynomials has a common root expressed in terms of the coefficients of the given polynomials." Assume a system of m polynomial equations in n variables. Different solution techniques are applied dependend on the values for m and n:

1. $m = 2$ and $n = 1$: The resultant, given by the determinant of the Sylvester matrix, is zero if and only if the two polynomials have common solutions.

2. $m = n+1$ and $n \geq 2$: The resultant, also called the Macaulay resultant, is calculated as the quotient of two determinants A and M. The system, which has to be homogenized first, has common solutions if A is equal to zero while M is nonzero. If M is also zero, the system still might have common solutions. This problem might be solved by introducing symbolic coefficients.

3. $m = n$: The resultant, also called u-resultant, is obtained by homogenizing the system, adding a so-called u-equation to the system and applying the Macaulay resultant.

The classical resultant-based solution techniques for the decision problem only address polynomial systems including equations but not inequations. In the application of DL systems as presented above, it is required to solve the decision problem for systems of polynomial equations and inequations. Thus, this work presents two different solution techniques. The first one is based on Gröbner bases while the second one relies on cylindrical algebraic decomposition. Gröbner bases are special polynomial systems which have the same set of solutions as the original polynomial system but can be solved applying various methods. A cylindrical algebraic decomposition of the space $\mathbb{R}^n$ can be described as a partition of $\mathbb{R}^n$ into connected subsets, called cells, on which each of the polynomials of the input polynomial system has constant sign.

The aim of this work is

- to search for significant solution techniques in order to solve the decision problem for a system of polynomial equations and inequations,
- to combine relevant information which is spread out widely in many different papers,
- to explain extremely complicated algebraic basics these solution techniques are relied on, and
- to present these solution techniques in an uniform way such that it is applicable to description logic applications.

This aim is reached by giving an algebraic overview in a general chapter and by introducing the solution techniques based on Gröbner bases and on cylindrical algebraic decomposition in two different chapters which can be read independently from each other. Thus, the chapters are arranged as follows. In Chapter 2, the mathematical background of linear and algorithmic algebra is presented. Chapter 3 analyzes solution techniques related to Gröbner bases in order to solve the decision problem for a set of polynomial equations and inequations. A method based on cylindrical algebraic decomposition to answer the decision question for a polynomial system is explored in Chapter 4. Chapter 6 summarizes the results of this work and inspects several performance studies presented in literature.

Chapter 2

Mathematical Background

This chapter introduces some of the basic concepts of linear and algorithmic algebra. It is mainly based on [Fis86], [Mis93], [CLO96] and [Yap00]. First, section 2.1 presents algebraic preliminaries, mainly in linear algebra. Section 2.2 is concerned with characteristics and properties of polynomials and polynomial rings. Subresultants and Sturm sequences are then discussed in Section 2.3. Finally, basics of ideal theory are given in Section 2.4.

In the following, $\mathbb{N}$, $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ denote the set of natural numbers $0, 1, 2, \cdots$, integers, rational numbers, reals, and complex numbers, respectively. It is assumed that the reader is familiar with matrix computations and analysis features such as derivatives.

2.1 Algebraic Preliminaries

First, the definition of relations, maps, groups, fields and vector spaces are recalled.

DEFINITION 2.1
Let A and B be two non-empty sets. A ***relation*** *R on A and B is defined as a subset of the Cartesian product $A \times B$. If $(a, b) \in R$ one says that a is related to b. This is also denoted by aRb or $a \sim b$.*

DEFINITION 2.2
Consider a binary relation R on one set A, i.e. $R \subset A \times A$. A relation R on A is called

- ***reflexive****, iff $\forall\, a \in A \Rightarrow (a, a) \in R$,*
- ***symmetric****, iff $\forall\, (a, b) \in R \Rightarrow (b, a) \in R$,*
- ***asymmetric****, iff $\forall\, (a, b) \in R \Rightarrow (b, a) \notin R$,*
- ***antisymmetric****, iff $\forall\, (a, b) \in R$ and $(b, a) \in R \Rightarrow a = b$,*
- ***transitive****, iff $\forall\, (a, b) \in R$ and $(b, c) \in R \Rightarrow (a, c) \in R$.*

DEFINITION 2.3
An ***ordering*** *on A is defined as a relation on A which is reflexive, antisymmetric and transitive. An irreflexive ordering is a relation on A which is irreflexive, asymmetric and transitive. An ordering is called a* ***quasi-ordering*** *if it is just reflexive and transitive, whereas an* ***irreflexive quasi-ordering*** *is one which is just irreflexive and transitive.*

DEFINITION 2.4
Two elements a, b of the set A are ***comparable*** *with respect to an ordering R iff either $(a, b) \in R$ or $(b, a) \in R$.*

DEFINITION 2.5
An ordering R on A is called ***total***, ***linear*** *or* ***simple*** *iff any two elements of A are comparable with respect to R. Otherwise, the ordering is called a* ***partial ordering***. *The tuple (A, R) is called a* ***partially ordered set*** *or* ***poset*** *in case R is a partial ordering. If R is a total ordering, (A, R) is called a* ***chain***.

For example, $(\mathbb{Z}, \leq)$ is a chain since any two elements a, b of A are always comparable: it is either $a \leq b$ or $b \leq a$. However, $(\mathbb{Z}, '\,divides')$ is a poset because the elements 2 and 7 are not comparable, for example. 2 does not divide 7 and vise versa. Therefore, the ordering 'divides' is a partial ordering.

DEFINITION 2.6
Let X and Y be two non-empty sets. Then, an instruction f or $f(x)$ is called a ***map*** *or a* ***mapping*** *if f assigns to each $x \in X$ a well defined element $f(x) \in Y$. This statement is denoted by:*

$$\begin{aligned} f: \; & X \to Y \\ & x \mapsto f(x) \end{aligned}$$

$f(x)$ is the ***image of x under the map f***. *The set X is called the* ***domain*** *of f, whereas the set Y is called the* ***range*** *of f.*

DEFINITION 2.7
Let $f : X \to Y$ be a map. Then f is called

- ***surjective***, *if $f(X) = Y$, i.e. for each $y \in Y$ there exists at least one $x \in X$ with $y = f(x)$,*
- ***injective***, *if for all $x, x' \in X$ with $f(x) = f(x')$ it holds that $x = x'$,*
- ***bijective***, *if f is surjective and injective.*

EXAMPLE 2.1
The map $f : \mathbb{R} \to \mathbb{R}_+$, $x \mapsto x^2$ is surjective, but not injective and the map $f : \mathbb{R}_+ \to \mathbb{R}$, $x \mapsto x^2$ is injective, but not surjective. But the map $f : \mathbb{R}_+ \to \mathbb{R}_+$, $x \mapsto x^2$ is surjective and injective and thus bijective.

DEFINITION 2.8
A ***group*** *is a tuple $(G, \circ)$ which consists of a non-empty set G and a binary operation on G, i.e. a map:*

$$\begin{aligned} \circ: \; & G \times G \to G \\ & (a, b) \mapsto a \circ b \end{aligned}$$

with the following properties (also called group axioms):

(G1) Associative law: $(a \circ b) \circ c = a \circ (b \circ c) \quad \forall\, a, b, c \in G.$

(G2) Identity element: There exists one identity element $e \in G$, for which the subsequent properties hold:

(G2a) $e \circ a = a \quad \forall\, a \in G$.

(G2b) Inverse element: For every element $a \in G$, there exists an inverse element, denoted by $a' \in G$, with $a \cdot a' = e$.

A group $(G, \circ)$ is called an ***Abelian group****, if the commutative law is valid:*

$$a \circ b = b \circ a \quad \forall\, a, b \in G.$$

For a ***semigroup****, the only group axiom that holds is (G1). A* ***monoid*** *is defined as a semigroup with identity, i.e. group axiom (G2a) is also valid. A* ***subgroup*** *of a group G is a non-empty subset G' iff $\forall\, a, b \in G'$ it holds that $a \circ b \in G'$.*

EXAMPLE 2.2

1. $(\mathbb{Z}, +)$, the set of integers with the operation addition, is an Abelian group. The identity element is 0, and the inverse element for any $z \in \mathbb{Z}$ is $z' = -z \in \mathbb{Z}$. Likewise $(\mathbb{Q}, +)$ and $(\mathbb{R}, +)$ are Abelian groups. However, $(\mathbb{N}, +)$ is not a group since condition (G2b) fails.

2. $(\mathbb{Q}^*, \cdot)$,where $\mathbb{Q}^* := \mathbb{Q} \backslash \{0\}$, i.e. the set of rational numbers without 0 with the operation multiplication, is an Abelian group. The identity element is 1, and the inverse element for any $q \in \mathbb{Q}^*$ is equal to $q' = \frac{1}{q} = q^{-1} \in \mathbb{Q}^*$. Equally, $(\mathbb{R}^*, \cdot)$ with $\mathbb{R}^* := \mathbb{R} \setminus \{0\}$ is an Abelian group. But $(\mathbb{Z}^*, \cdot)$ and $(\mathbb{N}^*, \cdot)$ with $\mathbb{Z}^* := \mathbb{Z} \setminus \{0\}$ and $\mathbb{N}^* := \mathbb{N} \setminus \{0\}$ are no groups, since condition (G2b) fails.

3. $(\mathbb{Z}_m, \oplus)$ is an Abelian group, whereas $(\mathbb{Z}_m^*, \odot)$ is not a group if $m = a \cdot b$ is a composite number with $a, b > 1$ and $\mathbb{Z}_m^* := \mathbb{Z}_m \setminus \{0\}$. $\mathbb{Z}_m$ denotes the set of residue classes modulo m, and $[r]$, $0 \leq r \leq m-1$, represents the residue class which consists of all numbers that have the same remainder when divided by m. $a \equiv b \, mod \, m$ iff a and b have the same remainder by division by m. $\oplus$ and $\odot$ are defined as follows: $[r_1] \oplus [r_2] := [r_1 + r_2]$ and $[r_1] \odot [r_2] := [r_1 \cdot r_2]$ with $a_1 + a_2 \equiv b_1 + b_2 \, mod \, m$ and $a_1 \cdot a_2 \equiv b_1 \cdot b_2 \, mod \, m$ for $a_1 \equiv b_1 \, mod \, m$ and $a_2 \equiv b_2 \, mod \, m$.

If it is clear which operation is meant, one shortly writes G instead of $(G, \circ)$ for a group. For groups with the operation addition $(+)$ or multiplication $(\cdot)$, the following conventions are adopted: In $a \cdot b$, one can omit the multiplication operator and just write ab. Instead of $a + (-b)$ one also writes $a - b$, as well as $\frac{a}{b}$ for ab^{-1}. For any interger $n \geq 1$, a^n denotes the multiplication of a with itself n times, i.e. $a^3 = aaa$, while a^{-n} is the inverse element of a^n.

DEFINITION 2.9

A ***ring*** *is a triple $(R, +, \cdot)$ which consists of a non-empty set R and two binary operations $+$ and $\cdot$ on R, called addition and multiplication, i.e. the two maps:*

$$+ : \; R \times R \to R \qquad\qquad \cdot : \; R \times R \to R$$
$$(a, b) \mapsto a + b \qquad\qquad (a, b) \mapsto a \cdot b$$

with the following properties:

(R1) $(R, +)$ *is an Abelian group. (The identity element is denoted by* 0*, and the inverse element with respect to addition for each* $x \in R$ *is represented by* $-x$*.)*

(R2) $(R, \cdot)$ *is a semigroup.*

(R3) Distributive law:

$$\begin{aligned} a \cdot (b + c) &= (a \cdot b) + (a \cdot c) \qquad \textit{and} \\ (a + b) \cdot c &= (a \cdot c) + (b \cdot c) \qquad \forall\, a, b, c \in R. \end{aligned}$$

The ring has an identity element, iff the semigroup with respect to multiplication has an identity element according to Definition 2.8, item $(G2a)$*. Further, the ring is called* ***commutative****, iff the multiplicative semigroup* $(R, \cdot)$ *is commutative, i.e.* $a \cdot b = b \cdot a$ $\forall\, a, b \in R$*.*
A ring is a ***field****, iff the multiplicative semigroup* $(R, \cdot)$ *can be replaced by a multiplicative Abelian group as follows:* $(R^*, \cdot)$ *is an Abelian group, where* R *is reduced to* $R^* := R \setminus \{0\}$*. (The identity element is* 1*, and the inverse element is then denoted by* x^{-1} *for every* $x \in R$*.) In general,* R *is replaced by the letter* K *in case of a field.*

If it is obvious, how addition and multiplication are defined on R or K to build a ring or field, one also writes R or K instead of $(R, +, \cdot)$ or $(K, +, \cdot)$, respectively. According to a general convention, multiplication ties stronger than addition. Therefore, the parentheses on the right hand side in item (R3) can be omitted. For a field element a and an integer $n \geq 1$, $n \cdot a$ means adding a to itself n times, i.e. $3 \cdot a = a + a + a$.

EXAMPLE 2.3

1. $(\mathbb{Z}, +, \cdot)$ is a ring with no zero divisors (see below).

2. $(\mathbb{Z}_m, \oplus, \odot)$ is a finite ring for any $m \geq 2$. $(\mathbb{Z}_m, \oplus, \odot)$ is called a residue class ring modulo m.

3. $(\mathbb{Z}_p, \oplus, \odot)$ is a finite field for any prime number p.

4. $(\mathbb{R}, +, \cdot)$ and $(\mathbb{Q}, +, \cdot)$ are examples for a field, whereas $(\mathbb{Z}, +, \cdot)$ is not a field since $(\mathbb{Z}^*, \cdot)$ with $\mathbb{Z}^* := \mathbb{Z} \setminus \{0\}$ is not an Abelian group as explained above.

DEFINITION 2.10
An element a*,* $a \neq 0$*, of a ring* R *is a* ***zero divisor*** *iff there exists an element* $b \in R$*,* $b \neq 0$*, such that* $a \cdot b = 0$*. An element* $a \in R$ *is called a* ***unit*** *iff there exists an element* $b \in R$ *such that* $a \cdot b = 1$*.*

A zero divisor divides 0 in parts different from 0. An example for a zero divisor is $a = 2$ in the residue class ring modulo 6 since $2 \cdot 3 = 0$ *mod* 6. In a field, every element that is different from zero is a unit since it has a uniquely determined inverse element. In the field of real numbers $\mathbb{R}$, $\mathbb{R} \setminus \{0\}$ represents the set of units, for example. In $\mathbb{Z}$, the units are $+1$ and -1.

DEFINITION 2.11
A ring R *is called an* ***integral domain*** *iff it has no zero divisor.*

DEFINITION 2.12
Let K be a field. A ***K-vector space*** *is a triple $(V, +, \cdot)$ which consists of a non-empty set V, a binary operation $+$ (addition), and a binary operation $\cdot$ (multiplication) with scalars such that:*

$$\begin{aligned} + :\ V \times V &\to V & \qquad \cdot :\ K \times V &\to V \\ (v, w) &\mapsto v + w & (\lambda, v) &\mapsto \lambda \cdot v \end{aligned}$$

with the following properties:

(V1) *$(V, +)$ is an Abelian group. (The identity element $\vec{0}$ is called* ***zero vector****, and the inverse element with respect to addition for each $v \in V$ is represented by its negative vector $-v$).*

(V2) *$\forall\, v, w \in V$ and $\lambda, \mu \in K$ the following conditions hold:*

$$\begin{aligned} (i) \quad & (\lambda + \mu) \cdot v = (\lambda \cdot v) + (\mu \cdot v), \\ (ii) \quad & \lambda \cdot (v + w) = (\lambda \cdot v) + (\lambda \cdot w), \\ (iii) \quad & (\lambda \cdot \mu) \cdot v = \lambda \cdot (\mu \cdot v), \\ (iv) \quad & 1 \cdot v = v. \end{aligned}$$

Elements of a vector space are called ***vectors****, also denoted by $\vec{v}$ or $\mathbf{v}$, while elements of K are called* ***scalars****.*

If K is known, one simply says vector space instead of K-vector space. Again, if it is clear which operations $+$ and $\cdot$ are applied for the vector space, one can also write V instead of $(V, +, \cdot)$. For $\lambda \cdot v$, where $\lambda \in K$ and $v \in V$, the multiplication operator can be omitted to yield the short form λv. Based on a usual convention, the tie for multiplication with scalars is stronger compared to addition in V and addition in K. Thus, some parentheses can be omitted. For a vector v and an integer $n \geq 1$, $n \cdot v$ denotes the vector obtained by adding v to itself n times, i.e. $3 \cdot v = v + v + v$.

EXAMPLE 2.4
The standard examples for K-vector spaces are the spaces K^n of n-tuples, where addition and multiplication is defined as follows:
$(x_1, \cdots, x_n) + (y_1, \cdots, y_n) := (x_1 + y_1, \cdots, x_n + y_n)$ and $\lambda(x_1, \cdots, x_n) := (\lambda x_1, \cdots, \lambda x_n)$. The null vector $\vec{0} = (0, \cdots, 0)$ is the identity element of the Abelian group $(K^n, +)$. The inverse element for each $\vec{v} \in K^n$ is represented by the negative vector $-\vec{v} = -\vec{v}$.

DEFINITION 2.13
Let V and W be two K-vector spaces and let $f : V \to W$ be a map. Then, f is called **K -linear** *iff for all $v, w \in V$ and $\lambda \in K$ the following conditions hold:*

(i) $f(v + w) = f(v) + f(w)$,

(ii) $f(\lambda \cdot v) = \lambda \cdot f(v)$.

One simply says ***linear*** *instead of K-linear, if it is clear which field is meant.*

Some linear maps for vector spaces have special names. They are defined in the sequel since they are used quite often.

DEFINITION 2.14
A linear map for vector spaces is also called a ***vector space homomorphism*** *or* ***homomorphism*** *for short. Let $f : V \to W$ be a homomorphism. Then, a homomorphism is further called a*

- ***monomorphism****, if f is injective,*
- ***epimorphism****, if f is surjective,*
- ***isomorphism****, if f is bijective,*
- ***endomorphism****, if $V = W$,*
- ***automorphism****, if $V = W$ and f is bijective.*

Next, a definition for the term real closed field is presented which needs some preparatory definitions first.

DEFINITION 2.15
An ***ordered field*** *K is a commutative field K togeher with a subset P, the set of positive elements, of K such that the following conditions hold:*

(1) $0 \notin P$.

(2) If $a \in K$, then either $a \in P$, $a = 0$, or $-a \in P$.

(3) P is closed under addition and multiplication: if $a, b \in P$, then so are $a + b$ and $a \cdot b$.

If a binary transitive relation $>$, defined as $a > b$ iff $(a - b) \in P$, is introduced in the ordered field K, Definition 2.15 can be described as: (1) $a = 0$ or $a > 0$ or $-a > 0$ and (2) $a > 0$ and $b > 0 \Longrightarrow a + b > 0$ and $a \cdot b > 0$. (It should be obvious that the binary transitive relation $<$ is defined in a similar way: $a < b$ iff $(b - a) \in P$. Thus, $-a > 0$ is equal to $a < 0$.)

DEFINITION 2.16
Let K be an ordered field. The ***sign function*** *evaluates the sign of an element $a \in K$ and is defined to be*

$$sign(a) = \begin{cases} +1 & \text{if } a > 0, \\ -1 & \text{if } a < 0, \\ 0 & \text{if } a = 0. \end{cases} \tag{2.1}$$

DEFINITION 2.17
A field K has ***characteristic p*** *iff there exists a smallest positive integer p such that $p \cdot 1 = 0$. If no such p exists, the characteristic of the field is defined to be zero.*

The characteristic of a field is either a prime number or 0. A field of characteristic zero is always infinite, whereas a field with positive characteristic can be either finite or infinite.

EXAMPLE 2.5
1. The fields $\mathbb{Q}$, $\mathbb{R}$, and $\mathbb{C}$ are fields of characteristic zero.

2. The finite field $\mathbb{Z}_p$ is a field of characteristic p. (For $\mathbb{Z}_p$ see Example (2.3), item 3.)

DEFINITION 2.18
A field K is called ***formally real*** *iff all equations in K of the form $\sum_{i=1}^{n} a_i^2 = 0$ are only true if every $a_i = 0$ for $a_i \in K$.*

This definition implies that K is formally real iff -1 is not a sum of squares of elements of K. Observe that every ordered field is formally real, and that a formally real field is necessarily of characteristic zero.

DEFINITION 2.19 (Real Closed Field)
An ordered field K is called ***real closed*** *if the following properties are satisfied:*

(1) Every positive element of K has a square root in K.

(2) Every polynomial $f(x) \in K[x]$ of odd degree has a root in K.

The next section introduces polynomials and polynomial rings, such as $K[x]$, in general. There also exists an alternative definition for a real closed field: A field K is real closed if K is formally real and no proper algebraic extension of K is formally real. $\mathbb{R}$ is the classical example of a real closed field.

2.2 Polynomials and Orderings

At the beginning of this section, polynomials and polynomial rings are defined. Then, properties of polynomials with respect to their roots are evaluated, and the Euclidean algorithm for the greatest common divisor is presented. This section concludes with orderings on polynomials.

Let $\mathbf{x}$ be in the following an abbreviation for $x_1, \cdots, x_n$. (It is not the vector $\mathbf{x}$ that is meant here.)

DEFINITION 2.20
A ***power product*** *or* ***term*** *over the variables $x_1, \cdots, x_n$ is a product of the form*

$$T = \prod_{i=1}^{n} x_i^{e_i} = x_1^{e_1} \cdots x_n^{e_n}, \tag{2.2}$$

where each $e_i \geq 0$ is an integer. The ***total degree*** *or simply* ***degree*** *$deg(T)$ of T is given by $\sum_{i=1}^{n} e_i$, and the* ***maximum degree*** *$mdeg(T)$ is defined as $max_{i=1}^{n} e_i$. $PP(\mathbf{x}) = PP(x_1, \cdots, x_n)$ denotes the set of power products over $\mathbf{x}$.*

DEFINITION 2.21
Let R be a ring. A ***monomial*** *is an expression of the form*

$$c \cdot T = c \cdot \prod_{i=1}^{n} x_i^{e_i} = c \cdot (x_1^{e_1} \cdots x_n^{e_n}), \tag{2.3}$$

where T is a power product and $c \in R \setminus \{0\}$ is called a ***coefficent****. The* ***total degree*** *or* ***degree*** *of a monomial is simply the total degree of its power product. Also, the* ***maximum degree*** *of a monomial is determined by the maximum degree of its power product. The* ***multidegree*** *of a monomial, is given by the vector of exponents $\mathbf{e} := (e_1, \cdots, e_n)$.*

DEFINITION 2.22
*A **polynomial** f in the variables $(x_1, \cdots, x_n)$ is uniquely defined as a finite linear combination of monomials with distinct power products:*

$$f(x) = \sum_{i=1}^{k} c_i \cdot T_i = \sum_{i=1}^{k} c_i \prod_{j=1}^{n} x_j^{e_{ij}} = \sum_{i=1}^{k} c_i \cdot x_1^{e_{i1}} \cdots x_n^{e_{in}}. \tag{2.4}$$

*The ring elements c_i are called the **coefficients** of f. A **univariate polynomial** is a polynomial in one variable, whereas a polynomial in more than one variable is called a **multivariate polynomial**. The **length** of a polynomial is the number of its monomials that does not have a coefficient of 0. The **total degree** or simply **degree** of a polynomial f, denoted by $deg(f)$, is defined as the maximum of the total degrees of the monomials in f. The **maximum degree** of a polynomial f, $mdeg(f)$, is the largest maximum degree of a monomial in f. By convention, $deg(0) = -\infty$. Let p be any power product over the variables $x_1, \cdots, x_n$ where the maximum degree is equal to 1. Then, the "p-degree" of a polynomial f, denoted by $deg_p(f)$, can be determined by simply viewing f as a polynomial in p while the rest of the variables are considered as part of the coefficents.*

A polynomial f is said to be an integer, rational, real, or complex polynomial, depending on whether R is equal to $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$. Throughout this whole work, it is assumed that any polynomial expression containing a finite linear combination of monomials is automatically simplified such that it only contains monomials with distinct power products. Non-simplified polynomial expressions are therefore also referred to as polynomials.

The definition of a polynomial ring with respect to two operations additon and multiplicaton is given in the following. Basically, the addition for two multivariate polynomials is performed by summing up the monomials of both polynomials and simplifying the received polynomial expression such that it only contains monomials with distinct power products. The multiplication is done by applying the distributive law that holds in any ring, followed by a simplification to a polynomial in the strong sense of Definition 2.22.

DEFINITION 2.23
*The **polynomial ring** over the variables $(x_1, \cdots, x_n)$, $n \geq 1$, with coefficients from a ring R is defined as the ring of the set of polynomials over R with the two operations addition and multiplication. Consider the two polynomials $f = \sum_{i=1}^{k} c_i \cdot x_1^{e_{i1}} \cdots x_n^{e_{in}}$ and $g = \sum_{j=1}^{l} c_j \cdot x_1^{e_{j1}} \cdots x_n^{e_{jn}}$. Then, addition and multiplication are defined as follows:*

$$f + g = \sum_{h=1}^{s} c_h \cdot x_1^{e_{h1}} \cdots x_n^{e_{hn}}, \tag{2.5}$$

where $max(k, l) \leq s \leq (k + l)$ and $c_h = a_i + b_j$ if there exists i, j such that $x_1^{e_{i1}} \cdots x_n^{e_{in}} = x_1^{e_{j1}} \cdots x_n^{e_{jn}}$, if not $c_h = a_i$ or $c_h = b_j$, respectively.

$$f \cdot g = \sum_{h=1}^{r} d_h \cdot x_1^{e_{h1}} \cdots x_n^{e_{hn}}, \tag{2.6}$$

where $r = k \cdot l$, $d_h = a_i \cdot b_j$ and $e_{ht} = e_{it} + e_{jt}$ for all $1 \leq t \leq n$, followed by a simplification, i.e. addition of monomials with equal power products. The polynomial ring is denoted by $R[x_1, \cdots, x_n] = R[\mathbf{x}]$. If R is a commutative ring with identity, then the polynomial ring $R[\mathbf{x}]$ is also a commutative ring with identity.

In the following, let R_0, R_1 be two rings such that $\mathbb{Z} \subseteq R_0 \subseteq R_1 \subseteq \mathbb{C}$. R_0 denotes the ring for the coefficients of a polynomial, whereas R_1 indicates the ring for the considered solutions of a polynomial. The definitions of polynomial functions, polynomial equation and inequations are given first.

DEFINITION 2.24
*Let $f = \sum_{i=1}^{k} c_i \cdot x_1^{e_{i1}} \cdots x_n^{e_{in}}$ be a polynomial in $R_0[x_1, \cdots, x_n]$. Then, the **polynomial function** corresponding to the polynomial f is defined by the following mapping:*

$$\begin{aligned} f:\ R_1 &\to R_1 \\ x &\mapsto f(x) := \sum_{i=1}^{k} c_i \cdot x_1^{e_{i1}} \cdots x_n^{e_{in}} \end{aligned}$$

DEFINITION 2.25
*Let $f \in R_0[x_1, \cdots, x_n]$ be a polynomial. Then, a **polynomial equation** is an expression of the form $f = 0$ while a **polynomial inequation** is any expression of the following forms: $f > 0$, $f \geq 0$, $f < 0$, or $f \leq 0$, and where $=, >, \geq, <, \leq$ are predicates over R_0.*

DEFINITION 2.26
*Consider the nonconstant polynomial $f \in R_0[\mathbf{x}]$ with coefficients in R_0. Then, a **solution**, **root**, or **zero** in R_1 of the polynomial f is defined as the solution of its corresponding **polynomial equation***

$$f = 0, \tag{2.7}$$

*which is any point $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n) \in R_1^n$ such that the polynomial function equals to zero: $f(\boldsymbol{\alpha}) = 0$. The **zero set** for the set of R_1 solutions is denoted by $ZERO_{R_1}(f)$. The subscript R_1 in $ZERO_{R_1}(f)$ is omitted in case $R_1 = \mathbb{R}$ and thus $ZERO(f)$ represents the set of real zeros. If R_1 is a field, the zero set is called an **algebraic set**.*

To name this set "algebraic" has historical reasons as already illustrated in the introduction.

DEFINITION 2.27
*Consider a Boolean combination of polynomial equations and inequations of the form $(f = 0)$, $(f > 0)$, or $(f \geq 0)$. The set of real solutions to such a Boolean combination is called a **semialgebraic set** or a **Tarski set**.*

EXAMPLE 2.6
An example for a Boolean combination of three polynomial equations and inequations is given by

$$[(f = 0) \wedge (g > 0)] \vee \neg(h \geq 0).$$

DEFINITION 2.28
*A **system of polynomial equations** in $n \geq 1$ variables is defined as follows:*

$$\left.\begin{aligned} f_1 &= 0 \\ f_2 &= 0 \\ &\vdots \\ f_p &= 0 \end{aligned}\right\} \quad \text{with } f_i \in R_0[x_1, \cdots, x_n] \text{ and } p \geq 1. \tag{2.8}$$

Equation (2.8) is the system of polynomial equations that corresponds to the polynomials $f_1, \cdots, f_p$. *A point* $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n) \in R_1^n$ *is called a* ***solution***, ***root*** *or* ***zero*** *of the system of equations 2.8 iff the corresponding polynomial functions are equal to zero,* $f_i(\boldsymbol{\alpha}) = 0$, $\forall i = 1, \cdots, p$. *The notation of the zero set for one polynomial,* $ZERO_{R_1}(f)$, *is extended to the notation for a set of polynomials* $P \subseteq R_0[\mathbf{x}]$ *such that* $ZERO_{R_1}(P) \subseteq R_1^n$ *denotes the* ***zero set*** *of* P *which is defined as the set of common solutions in* R_1 *of the corresponding system of polynomial equations. The subscript* R_1 *in* $ZERO_{R_1}(f)$ *is omitted in case* $R_1 = \mathbb{R}$ *and thus* $ZERO(P)$ *represents the set of real zeros.*

Definition 2.28 can be extended to a system of polynomial equations and inequations in $n \geq 1$ variables. The polynomial equations and inequations are now specified by $f_i \, \rho \, 0$, where $f_i \in R_0[x_1, \cdots, x_n]$ and $\rho \in \{<, \leq, =, \geq, >\}$. A point $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n) \in R_1^n$ is called a **solution** of the system of polynomial equations and inequations iff $f_i(\boldsymbol{\alpha}) \, \rho \, 0$ $\forall i = 1, \cdots, p$. $SOL_{R_1}(S) \subseteq R_1^n$ indicates the set of common solutions of S, where S can be any set of polynomial equations and inequations. Here, S can also be a set of polynomial equations only. In case R_1 is equal to $\mathbb{R}$ the set of real solutions is a semialgebraic set as defined in Definition 2.27 since polynomial inequations containing the predicates $\leq$ and $<$ can be transformed to polynomial equations and inequations containing only the predicates $>, \geq$ and $=$.

In the following a **polynomial system** denotes a polynomial system of equations and inequations in general, which means that it can also be only a polynomial system of equations. Also, a polynomial system is called an integer, rational, real, or complex polynomial system, depending on whether the ring of the coefficients is equal to $\mathbb{Z}$, $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$.

DEFINITION 2.29
Let S *be a set of polynomial equations and inequations defining a polynomial system. The polynomial system is said to be* ***satisfiable*** *or* ***solvable*** *in* R_1 *iff it has at least one common zero in* R_1, *i.e.* $|SOL_{R_1}(S)| \neq \{\}$. *Otherwise it is called* ***unsatisfiable*** *or* ***unsolvable***. *The polynomial system is said to be* ***finitely solvable*** *iff it is solvable and* S *has finitely many zeros.*

The definition of the term "algebraically closed" is obviously based on the origin of the name "algebra" as described in the introduction. The Fundamental Theorem of Algebra then states that the complex numbers are a field which is algebraically closed.

DEFINITION 2.30
A field K *is* ***algebraically closed*** *iff every non-constant polynomial in* $K[x]$ *has a root in* K.

THEOREM 2.1 *The* ***Fundamental Theorem of Algebra*** *states that every nonconstant polynomial* $f \in \mathbb{C}[\mathbf{x}]$ *has a root* $\boldsymbol{\alpha} \in \mathbb{C}$. *This means that* $\mathbb{C}$ *is algebraically closed.*

Some interesting properties of univariate polynomials concerning their roots are presented now. Similar properties of multivariate polynomials are considered in Chapter 3, where Gröbner Bases are discussed to solve the decision problem.
Consider the complex polynomial $f = \sum_{i=0}^{n} a_i \cdot x_i, \quad a_n \neq 0$, $n \geq 1$. Let f have exactly n distinct complex roots, $\alpha_1, \cdots, \alpha_n \in \mathbb{C}$. The factorized form of the polynomial f can be

written as follows:

$$f = a_n \prod_{i=0}^{n} (x - \alpha_i)^{m_i},$$

where $m_i \geq 1$, $m_i \in \mathbb{N}$.

DEFINITION 2.31
Let $f = a_n \prod_{i=0}^{n} (x - \alpha_i)^{m_i}$ be a complex polynomial in factorized form. Then, the root α_i is a root of ***multiplicity*** $\mathbf{m_i}$ *of f. Alternatively, α_i is called an* $\mathbf{m_i}$***-fold root*** *of f. A root is* ***simple*** *or* ***multiple*** *according to whether $m_i = 1$ or $m_i \geq 2$. f is said to be* ***"square" free*** *iff it has no multiple roots.*

This implies immediately that if α is a root of f of multiplicity $m \geq 1$ then α is a root of its first derivative f' of multiplicity $m - 1$.

COROLLARY 2.2 *Let K be an arbitrary field and let $f \in K[x]$ be a nonzero polynomial. Then, the polynomial $\hat{f}$*

$$\hat{f} := \frac{f}{GCD(f, f')} \tag{2.9}$$

is square free and contains exactly the distinct roots of f.

The Corollary 2.2 implies how to find a univariate square free polynomial $\hat{f}$ for every polynomial $f \in K[x]$. For this task, the greatest common divisior, GCD, is applied. Before presenting the definition of the greatest common divisor, the term "divide" needs to be explained. Every $f \in K[x]$ can be expressed with respect to a nonzero polynomial $g \in K[x]$ as follows:

$$f = qg + r, \tag{2.10}$$

where $q, r \in K[x]$ are unique, and either $r = 0$ or $deg(r) < deg(g)$. r and q are called **remainder** and **quotient**, respectively. In the case of $r = 0$ it is said that polynomial g divides polynomial f.

DEFINITION 2.32
The ***greatest common divisor*** *of the polynomials $f_1, \cdots, f_p \in K[x]$ is a polynomial h with the following properties:*

(1) h divides $f_1, \cdots, f_p$.

(2) If g is another polynomial which divides $f_1, \cdots, f_p$, then g divides h.

The greatest common divisor is denoted by $gcd(f_1, \cdots, f_p)$ and is unique up to multiplication by a nonzero constant in K. Furthermore, if $p \geq 3$, then $gcd(f_1, \cdots, f_p) = gcd(f_1, gcd(f_2, \cdots, f_p))$.

The greatest common divisor is computed with the Euclidean algorithm, which is presented in Algorithm 2.1. Algorithm 2.1 computes the greatest common divisor of two univariate polynomials based on the fact that $GCD(f, g) = GCD(g, r)$, where f is expressed as in Equation (2.10). If $r \neq 0$, the polynomial division is applied continuously until the remainder equals zero.

Algorithm 2.1 GCD(f_1, f_2)

Input: Two polynomials f_1 and f_2 in $K[x]$.
Output: The greatest common divisor, $gcd(f_1, f_2)$, of f_1 and f_2.
$h := f_1$; $s := f_2$;
while $(s \neq 0)$ **do**
  $(x, r) := DIVISON(h, s)$; {x is not used}
  $h := s$;
  $s := r$;
end while
return h ;

Algorithm 2.2 DIVISION(f_1, f_2)

Input: Two polynomials f_1 and f_2 in $K[x]$.
Output: The quotient q, and the remainder r of the division.
$r := f_1$; $q := 0$;
while $(r \neq 0)$ **and** $(hmono(f_2)\ divides\ hmono(r))$ **do**
  $q := q + hmono(r)/hmono(f_2)$**;**
  $r := r - (hmono(r)/hmono(f_2)) \cdot f_2$**;**
end while
return (q, r) **;**

Algorithm 2.1 invokes the division algorithm DIVISON, Algorithm 2.2, which calculates the quotient q and the remainder r for two input polynomials f and g as defined in Equation (2.10).

The head monomial $hmono(f)$ for a univariate polynomial f denotes that monomial in f which has the largest degree. According to Definition (2.21), a monomial consists of a power product and a coefficient. The monomial $hmono(g)$ *divides* the monomial $hmono(f)$ if and only if the exponent of the power product of the monomial $hmono(g)$ is less or equal to the exponent of the power product of the monomial $hmono(f)$ and if and only if the coefficient of the monomial $hmono(g)$ divides the coefficient of the monomial $hmono(f)$ in the field K, the field of coefficients of the polynomial ring $K[x]$.

The GCD algorithm therminates because the degree of the remainder gets smaller in each step of the loop. This is true since Equation (2.10) with either $r = 0$ or $deg(r) < deg(g)$ is always valid throughout the algorithm. In case the GCD algorithm is called with two polynomials f and g, GCD(g,f), where $deg(g) < deg(f)$ holds, the DIVISON algorithm returns $(0, f)$ and the arguments are switched for the next call of the DIVISON algorithm. The GCD algorithm stops when the remainder equal zero and the greatest common divisor of the polynomials f and g is given by the divisor of the last computed polynomial division.

The following theorem states an upper and lower bound of an interval within which all real zeros of a polynomial can be found. These bounds will be used later on for real root isolation in Chapter (4). The subsequent corollary denotes the minimum separation between any two roots of a polynomial.

THEOREM 2.3 *Let $f \in \mathbb{Z}[x]$ be a univariate integer polynomial:*

$$f = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0,$$

for which different norms are defined as follows:

$$\|f\|_1 = |a_n| + |a_{n-1}| + \cdots + |a_0| \tag{2.11}$$

$$\|f\|_2 = (a_n^2 + a_{n-1}^2 + \cdots + a_0^2)^{\frac{1}{2}} \tag{2.12}$$

$$\|f\|_\infty = max\,(|a_n|, |a_{n-1}|, \cdots, |a_0|). \tag{2.13}$$

Then all real roots of f are in the intervals $(-\|f\|_1, \|f\|_1)$, $[-\|f\|_2, \|f\|_2]$ and $(-1 - \|f\|_\infty, 1 + \|f\|_\infty)$. Also for every nonzero real root α of f the following holds:

$$|\alpha| > \frac{1}{1 + \|f\|_\infty}. \tag{2.14}$$

DEFINITION 2.33
If f is an complex polynomial of degree n, not necessarily square free, its minimal root seperation $sep(f)$ is defined by:

$$sep(f) := min_{1 \le i < j \le k} |\alpha_i - \alpha_j|, \tag{2.15}$$

where the distinct roots of f are $\alpha_1, \cdots, \alpha_k \in \mathbb{C}$. If f has less than two distinct roots, then $sep(f) = \infty$.

COROLLARY 2.4 *Rump and Schwartz obtained a bound for the minimal root seperation as given in [Yap00] by:*

$$sep(f) > [2 \cdot n^{(n/2)+2}(\|f\|_\infty + 1)]^{-1}. \tag{2.16}$$

The following Corollary explaines why every polynomial $f \in \mathbb{R}[x]$ with an odd degree has at least one real root.

COROLLARY 2.5 *The nonreal roots of $f \in \mathbb{R}[x]$ appear in conjugate pairs. Let $\bar{\alpha} = a - -i \cdot b$ denote the conjugate complex number of $\alpha = a + i \cdot b$, where $i^2 = -1$. Then, the following two conditions hold since $(x - \alpha)(x - \bar{\alpha})$ is a real polynomial:*

(1) f can be written as a product of real factors that are linear or quadratic.

(2) If $deg(f) = n$ is odd, then f has at least one real root.

For multivariate polynomials, a precedence of the involved variables and an admissible ordering has to specified. Admissible orderings on power products are discussed first. Then, they are then extended to a quasi-ordering for monomials and polynomials. In the following "A" in $\underset{A}{\leq}$ is a placeholder for any admissible ordering.

DEFINITION 2.34
Let $PP = PP(x_1, \cdots, x_n)$ be the set of power products of $\mathbf{x}$ as defined in Definition 2.20. A partial ordering $\underset{A}{\leq}$ on PP is ***compatible*** *iff for all p, q, $r \in PP$, $p \underset{A}{\leq} q$ implies $rp \underset{A}{\leq} rq$.*

DEFINITION 2.35
A total ordering $\underset{A}{\leq}$ *on* PP *is said to be* ***semiadmissible*** *iff it is compatible. It is* ***admissible*** *iff it is semiadmissible and for all* $p \in PP$ *it follows that* $1 \underset{A}{\leq} p$.

There are three important admissible orderings on the set of power products: the **(pure) lexicographic ordering** $\underset{lex}{\leq}$, the **(total) degree ordering** $\underset{tot}{\leq}$, and the **reverse lexicographic ordering** $\underset{rev}{\leq}$. These orderings can be uniquely defined once a precedence of the variables $x_1, \cdots, x_n$ is chosen. Typically, the index number defines their precedence: $x_1 < x_2 < \cdots < x_n$, where $<$ is an total irreflexive ordering. It should be obvious that the admissible orderings can be ordered in reverse direction, which is then denoted by $\underset{lex}{\geq}$, $\underset{tot}{\geq}$, and $\underset{rev}{\geq}$, respectively.

DEFINITION 2.36
Let $p = x_1^{d_1} x_2^{d_2} \cdots x_n^{d_n}$ *and* $q = x_1^{e_1} x_2^{e_2} \cdots x_n^{e_n}$ *be two power products in* $PP(x_1, \cdots, x_n)$, *where the exponents can be expressed as vectors* $\mathbf{d} = (d_1, \cdots, d_n)$ *and* $\mathbf{e} = (e_1, \cdots, e_n)$. *Then,*

(a) $p \underset{lex}{\geq} q$ *if* $p = q$ *or else, for the smallest* i $(1 < i < n)$ *such that* $d_i \neq e_i$, $d_i > e_i$, *i.e the first nonzero component in the vector* $\mathbf{d} - \mathbf{e} = (d_1 - e_1, d_2 - e_2, \cdots, d_n - e_n)$ *is positive,*

(b) $p \underset{tot}{\geq} q$ *if* $deg(p) > deg(q)$ *or else,* $p \underset{lex}{\geq} q$,

(c) $p \underset{rev}{\geq} q$ *if* $deg(p) > deg(q)$ *or else, if* $deg(p) = deg(q)$, *for the largest* i $(1 < i < n)$ *such that* $d_i \neq e_i$, $d_i < e_i$, *i.e the last nonzero component in the vector* $\mathbf{d} - \mathbf{e}$ *is negative.*

For all admissible orderings it holds that $p \underset{A}{>} q$ *iff* $p \underset{A}{\geq} q$ *and* $p \neq q$.

COROLLARY 2.6 *Let* $PP(x, y, z)$ *be the set of power products on the variables* x, y, z *with the total irreflexive ordering* $z < y < x$. *Then, the admissible orderings for* $\underset{lex}{<}$, $\underset{tot}{<}$ *and* $\underset{rev}{<}$ *are as follows:*

(a) $1 \underset{lex}{<} z \underset{lex}{<} z^2 \cdots \underset{lex}{<} y \underset{lex}{<} yz \cdots \underset{lex}{<} y^2 \cdots \underset{lex}{<} x \underset{lex}{<} xz \cdots \underset{lex}{<} xy \cdots \underset{lex}{<} x^2 \cdots$,

(b) $1 \underset{tot}{<} z \underset{tot}{<} y \underset{tot}{<} x \underset{tot}{<} z^2 \underset{tot}{<} yz \underset{tot}{<} y^2 \underset{tot}{<} xz \underset{tot}{<} xy \underset{tot}{<} x^2 \underset{tot}{<} \cdots$,

(c) $1 \underset{rev}{<} z \underset{rev}{<} y \underset{rev}{<} x \underset{rev}{<} z^2 \underset{rev}{<} yz \underset{rev}{<} xz \underset{rev}{<} y^2 \underset{rev}{<} xy \underset{rev}{<} x^2 \underset{rev}{<} \cdots$.

EXAMPLE 2.7
Let $f = 4xy^2z + 4z^2 - 5x^3 + 7x^2z^2$ be a polynomial in $K[x, y, z]$ and let $x > y > z$. Then, the monomials in f would have to be reordered with respect to the three mentioned addmissible orderings:

(a) $\underset{lex}{>}$: $f = -5x^3 + 7x^2z^2 + 4xy^2z + 4z^2$,

(b) $\underset{tot}{>}$: $f = 7x^2z^2 + 4xy^2z - 5x^3 + 4z^2$,

(c) $\underset{rev}{>}$: $f = 4xy^2z + 7x^2z^2 - 5x^3 + 4z^2$,

In case of the reverse lexicographic ordering, $4xy^2z \underset{rev}{>} 7x^2z^2$ holds according to Definition 2.36, since both monomials have the same degree, which is 4, and the first exponent they differ is the one for the variable x, which is 1 for $4xy^2z$ and 2 for $7x^2z^2$ and which is thus smaller for the first monomial than for the second one.

The admissible ordering $\underset{A}{\leq}$ for power products is now extended to a total quasi-ordering for monomials and polynomials.

DEFINITION 2.37
Let $c \cdot p$ and $d \cdot q$ be two monomials with $c, d \in K$ and $p, q \in PP$, where K is an arbitrary field and PP is the set of power products. Then, $c \cdot p \underset{A}{\leq} d \cdot q$ if $p \underset{A}{\leq} q$.

DEFINITION 2.38
*Let $f = c_1p_1 + c_2p_2 + \cdots + c_kp_k$ and $g = d_1q_1 + d_2q_2 + \cdots + d_lq_l$ be two polynomials, written each as a sum of monomials with distinct p_i's and q_i's, respectively. The **term sequence** of f is given by $\bar{f} = (p_1, \cdots, p_k)$, where $p_1 \underset{A}{>} p_2 \underset{A}{>} \cdots \underset{A}{>} p_k$. Then, $f \underset{A}{\leq} g$ iff in their term sequences $(p_1, \cdots, p_k)$ and $(q_1, \cdots, q_l)$ either*

(1) $k \leq l$ and $p_i = q_i \; \forall \, i = 1, \cdots, k$ or

(2) for some $i \leq min\{k, l\}$, $p_i < q_i$ and $p_j = q_j$ for $j = 1, \cdots, i-1$.

To compare two polynomials, one scans through the their term sequences componentwise until one finds the first component pair that differs. The polynomials to which the smaller component belongs is defined to be the smaller polynomial. If one of the term sequences runs out of components before a differing component pair could be found, it is immediately defined to be the smaller polynomial.

With respect to a given admissible ordering $\underset{A}{\leq}$, the head term, head monomial and the head coefficient of a multivariate polynomial can be determined.

DEFINITION 2.39
*Let $f = \sum_{i=1}^{k} c_i \prod_{j=1}^{n} x_j^{e_{ij}} = \sum_{i=1}^{k} c_i \cdot x_1^{e_{i1}} \cdots x_n^{e_{in}}$ be a polynomial in $R[x_1, \cdots, x_n]$. Then, the **head monomial** of f, denoted by $hmono(f)$, is represented by the monomial that contains the $\underset{A}{\leq}$-largest power product or term in f. The **head coefficient** of f, $hcoef(f)$, and the **head term** of f, denoted by $hterm(f)$, refers to the coefficient and the term associated to the head monomial of f, respectively. Thus, $hmono(f) = hcoef(f) \cdot hterm(f)$. Let $F \subseteq R[x_1, \cdots, x_n]$ be a set of polynomials, $hterm(F)$ is the extension of $hterm(f)$ to the set $hterm(F) = \{hterm(f) \mid f \in F\}$. The **multidegree** of f, $multideg(f)$, is given by the vector of exponents $\mathbf{e}_i := (e_{i1}, \cdots, e_{in})$ of the $\underset{A}{\leq}$-largest power product in f, i.e. of $hterm(f)$.*

EXAMPLE 2.8
Assume $x > y$ and consider the polynomial $f = -2x^2y + x + 1$. Then $hmono(f) = -2x^2y$, $hterm(f) = x^2y$ and $hcoef(f) = -2$. In this example the evaluation of the head monomial,

head term and head coefficient does not depend on the selection of the admissible ordering as presented in Example 2.6. If the polynomial $g = 2x + y^3$ is inspected, the evaluation actually depends on the selection of the admissible ordering:

- $\underset{lex}{\leq}$: $hmono(g) = 2x$, $hterm(g) = x$, $hcoef(g) = 2$, $multideg(g) = (1, 0)$,
- $\underset{tot}{\leq}$: $hmono(g) = -y^3$, $hterm(g) = y^3$, $hcoef(g) = -1$ $multideg(g) = (0, 3)$.

The next definition introduces the reductum of a polynomial. The reductum is the polynomial received from the original one by cutting off the head monomial. Applying the reductum recursively to a polynomial will result in a set of polynomials, called the set of reducta.

DEFINITION 2.40
The ***reductum****, $red(f)$, with respect to a specific ordering is defined by: $red(f) = f - hmono(f)$. The* $\mathbf{i^{th}}$ ***reductum*** *of f, written $red^i(f)$ is received via induction on i: $red^0(f) = f$ and $red^{i+1}(f) = red(red^i(f))$. The* ***set of reducta*** *is defined as follows:*

$$RED(f) = \{ \bigcup_{i=1}^{deg(f)} red^i(f) \mid red^i(f) \neq 0\} \tag{2.17}$$

EXAMPLE 2.9
Consider the following polynomial in the variable z:

$$f(x, y, z) = (y^2 + x^2 - 1) \cdot z^3 + (x - 1) \cdot z^2 + ((x - 1)^2 + y^2) \cdot z^0.$$

Then, the set of reducta of the polynomial is

$$RED(f) = \{(y^2+x^2-1)\cdot z^3+(x-1)\cdot z^2+(x-1)^2+y^2, (x-1)\cdot z^2+(x-1)^2+y^2, (x-1)^2+y^2\}.$$

2.3 Pseudo-Divison, PRS, Subresultants, and Sturm Sequences

This section introduces important mathematical tools such as division for the multivariate case, polynomial remainder sequences, subresultants, principal subresultant coefficients and Sturm sequences.

The most important tool to analyze univariate polynomials is the greatest common divisor (GCD), which has been defined in Definition 2.32. It is possible to determine the number of common zeros of two given polynomials as well as the number of distinct zeros of one polynomial using the greatest common divisor. In general, the number of zeros (including multiplicity) of a single univariate polynomial is equal to its degree. The number of common zeros of two univariate polynomials can be evaluated from the degree of their greatest common divisor.

LEMMA 2.7 *The number of common zeros of the two polynomials $f, g \in K[x]$ is*

$$deg(GCD(f, g)).$$

This lemma becomes obvious from the definition of the greatest common divisor and the fact that the number of zeros (including multiplicity) of a univariate polynomial is equal to its degree. This is also true for multivariate polynomials which are considered as polynomials in one variable, for example x_n, while all other occuring variables, $x_1, x_2, \ldots, x_{n-1}$, are regarded as part of the coefficents. In this case the polynomial ring of these polynomials is often denoted in the form $K[x_1, \ldots, x_{n-1}][x_n]$ instead of $K[x_1, \ldots, x_{n-1}, x_n]$.

In the following, let f' denote the first derivative of f.

LEMMA 2.8 *The number of distinct zeros of a polynomial $f \in K[x]$ is*

$$deg(f) - deg(GCD(f, f')).$$

EXAMPLE 2.10
Given the polynomial $f(x) = 6 \cdot (x-2)^4 \cdot (x+4)^1 \cdot (x-3)^2$, its degree is 7. Written in this factorized form, the number of distinct zeros can be evaluated to be 3. The derivative of $f(x)$ is $f'(x) = 6 \cdot 4 \cdot (x-2)^3 \cdot 1 \cdot 2 \cdot (x-3)^1$. Obviously, the greatest common divisor can be computed to $GCD(f, f') = 6 \cdot (x-2)^3 \cdot (x-3)^1$, which has a degree of 4. Hence the number of distinct zeros results in 3.

The subsequent polynomial divisions, obtained by applying the Euclidean algorithm to two univariate polynomials as given by Algorithm 2.1, can be outlined as follows:

$$\begin{aligned}
f_1 &= q_1 \cdot f_2 + f_3, && deg(f_3) < deg(f_2) \\
f_2 &= q_2 \cdot f_3 + f_4, && deg(f_4) < deg(f_3) \\
&\vdots && \\
f_{p-2} &= q_{q-2} \cdot f_{p-1} + f_p, && deg(f_p) < deg(f_{p-1}) \\
f_{p-1} &= q_{q-1} \cdot f_p + 0. &&
\end{aligned} \tag{2.18}$$

The sequence $(f_1, f_2, \ldots, f_p)$ is called a **polynomial remainder sequence (PRS)**.

So far univariate polynomials have been considered. The case is slightly different for multivariate polynomials. Multivariate polynomials are considered here to be elements of the polynomial ring $K[x_1, \ldots, x_{n-1}][x_n]$. This poses the difficulty that the number of zeros and the locations of the zeros vary for different coefficient points $\alpha = (\alpha_1, \ldots, \alpha_{n-1}) \in K^{n-1}$. Therefore, the greatest common divisor of two multivariate polynomials also depends on α. The GCD-calculation for multivariate polynomials is performed with the Euclidean Algorithm using **pseudo division**, which is explained in the following definition, and which is implemented by Algorithm 2.3. The degree of a polynomial with respect to a certain variable x_i is denoted by $deg(f_1)_{x_i}$. The head coefficient $hcoef(f_1)_{x_i}$ can be defined respectively.

DEFINITION 2.41
Let $f_1, f_2 \in K[x_1, \ldots, x_{n-1}][x_n]$ be two polynomials with $deg(f_1) = m$, $deg(f_2) = h$, $hcoef(f_1) = a_m$, and $hcoef(f_2) = b_h$ such that:

$$\begin{aligned}
f_1 &= a_m \cdot x_n^m + \cdots + a_1 \cdot x_n + a_0 \quad \textit{and} \\
f_2 &= b_h \cdot x_n^h + \cdots + b_1 \cdot x_n + b_0
\end{aligned}$$

with a_i *and* $b_j \in K[x_1, \ldots, x_{n-1}]$ *for* $0 \leq i \leq m$ *and* $0 \leq j \leq h$. *Assume furthermore* $h \leq m, f_2 \neq 0$. *Then, there exist two polynomials* q *and* r *in* $K[x_1, \ldots, x_{n-1}, x_n], s \geq 0$ *such that:*

$$b_h^s \cdot f_1 = q \cdot f_2 + r \qquad (r = 0 \quad or \quad deg(r)_{x_n} < h). \tag{2.19}$$

r *is called the* ***pseudo remainder****, denoted by PREM(* f_1, f_2*).*

Algorithm 2.3 PSEUDO-DIVISION(f_1, f_2)

Input: Two polynomials f_1 and f_2 in $K[x_1, \cdots, x_{n-1}][x_n]$.
Output: The quotient q and the pseudo remainder r.
$r := f_1$; $q := 0$; $h := deg(f_2)_{x_n}$;
$b_h := hcoef(f_2)_{x_n}$;
while $(r \neq 0)$ **and** $(deg(r)_{x_n} \geq h)$ **do**
  $c := hcoef(r)_{x_n}$**;**
  $d := deg(r)_{x_n}$**;**
  $r := b_h \cdot r - c \cdot f_2 \cdot x_n^{d-h}$**;**
  $q := b_h \cdot q + c \cdot x_n^{d-h}$**;**
end while
return (q, r) **;**

Pseudo division can be easily described as ordinary one-variable polynomial division for polynomials in x_n with coefficients in $K[x_1, \ldots, x_{n-1}]$, followed by canceling denominators in each division step by multiplying with the head coefficient of the divisor. The idea is to eliminate the head coefficients of r in each iteration of the while loop. The variable s in Equation 2.19 is not returned by the algorithm, but represents the counter of the while loop.

Example 2.11 explains pseudo divison using two polynomials in the variable y with coefficients in the variable x.

EXAMPLE 2.11 (Pseudo division)
Given the following polynomials

$$f_1 = x^2y^3 - y$$
$$f_2 = x^3y - 2$$

the pseudo division of f_1 by f_2 is determined as follows:
Init: $r = x^2y^3 - y$, $q = 0$, $h = 1$, $b_h = x^3$
Loopstep $s = 1$: $c = x^2$, $d = 3$,
$r = x^3 \cdot (x^2y^3 - y) - x^2 \cdot (x^3y - 2) \cdot y^{3-1} = 2x^2y^2 - x^3y$,
$q = x^3 \cdot 0 + x^2 \cdot y^{3-1} = x^2y^2$
Loopstep $s = 2$: $c = 2x^2$, $d = 2$,
$r = x^3 \cdot (2x^2y^2 - x^3y) - 2x^2 \cdot (x^3y - 2) \cdot y^{2-1} = 2x^5y^2 - x^6y - 2x^5y^2 + 4x^2y = (4x^2 - x^6)y$,
$q = x^3 \cdot (x^2y^2) + 2x^2 \cdot y^{2-1} = x^5y^2 + 2x^2y$
Loopstep $s = 3$: $c = (4x^2 - x^6)$, $d = 1$,
$r = x^3 \cdot ((4x^2 - x^6)y) - (4x^2 - x^6) \cdot (x^3y - 2) \cdot y^{1-1} = 4x^5y - x^9y - 4x^5y + 8x^2 + x^9y - 2x^6$
$= 8x^2 - 2x^6$,
$q = x^3 \cdot (x^5y^2 + 2x^2y) + (4x^2 - x^6) \cdot y^{1-1} = x^8y^2 + 2x^5y + 4x^2 - x^6$

A polynomial remainder sequence (PRS) for univariate polynomials can now be extended to a polynomial remainder sequence for multivariate polynomials. First, the similarity of two polynomials is explained.

DEFINITION 2.42
Two polynomials in $K[x_1, \ldots, x_{n-1}][x_n]$ *are* ***similar****, denoted by* $f_1(x_n) \sim f_2(x_n)$*, iff there exist* $a, b \in K[x_1, \ldots, x_{n-1}]$ *such that* $af_1(x_n) = bf_2(x_n)$*.*

DEFINITION 2.43 (PRS for multivariate polynomials)
Let $f_1, f_2 \in K[x_1, \ldots, x_{n-1}][x_n]$ *with* $deg(f_1) \geq deg(f_2)$ *in* x_n*. The sequence* $f_1, \ldots, f_p$ *is a* ***polynomial remainder sequence (PRS)*** *for* f_1 *and* f_2 *iff:*

(i) $\forall i = 3, \ldots, p \quad f_i \sim PREM(f_{i-2}, f_{i-1})$

(ii) the sequence terminates with $PREM(f_{p-1}, f_p) = 0$*.*

The polynomial remainder sequence (PRS) is unique up to similarity: $GCD(f_1, f_2) \sim \ldots \sim GCD(f_{p-1}, f_p) \sim f_p$. Basically, one can distinguish between two sequences:

- Euclidean PRS (EPRS):

$$\begin{aligned} f_i = PREM(f_{i-2}, f_{i-1}) \neq 0 \\ PREM(f_{p-1}, f_p) = 0 \end{aligned} \tag{2.20}$$

- Primitive PRS (PPRS):

$$\begin{aligned} f_i = PRIM(PREM(f_{i-2}, f_{i-1})) \neq 0 \\ PREM(f_{p-1}, f_p) = 0 \end{aligned} \tag{2.21}$$

where PRIM denotes the primitive part of the polynomial in which all the common factors of the coefficients have been removed. Both, EPRS and PPRS, have a high computational complexity due to an exponential coefficient growth for EPRS and due to GCD-calculations for the coefficients in the case of PPRS, respectively. The so called Subresultant Polynomial Remainder Sequence (SPRS) is a tradeoff between EPRS and PPRS. The SPRS is not explained in detail here because it would exceed the scope of this work. Please refer to [Yap00] or to [Mis93] for further information.

Now, the concept of subresultants and principle subresultant coefficients is explained. They are an important mathematical tool to perform projections as it is required for the cylindrical algebraic decomposition. Let $U = K[x_1, \ldots, x_{n-1}]$ throughout this part.

DEFINITION 2.44
Let a set of polynomials be defined by $f_i = \sum_{j=0}^{m_i} a_{ij} \cdot x^j \in U[x_n], i = 1, \ldots, p$*. The* ***matrix associated*** *with* $f_1, \ldots, f_p$ *is obtained by applying the matrix-operator on a*

vector of polynomials:

$$MATRIX((f_1, \ldots, f_p)) = [m_{st}] = [a_{s(l-t)}] \tag{2.22}$$

where $l = 1 + max_{1 \leq i \leq p}(m_i)$, $1 \leq s \leq p$, *and* $1 \leq t \leq l$. *Remember that* $[m_{st}]$ *is a short form of the following matrix:*

$$[m_{st}] = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1t} \\ m_{21} & m_{22} & \cdots & m_{2t} \\ \vdots & \vdots & & \vdots \\ m_{s1} & m_{s2} & \cdots & m_{st} \end{bmatrix}. \tag{2.23}$$

A new pair of indices (s, t) *is introduced for the matrix, since the index for the rows and the index for the columns start at* 1, *whereas the index* j *in the definition of the polynomial starts at* 0.

This simply means that the coefficients of the polynomials are written from the left to the right into a matrix starting with the highest degree of the polynomials first. The number of matrix rows corresponds to the number of polynomials. Pseudo division can also be achieved by matrix operations on a certain matrix containing the coefficients of the two polynomials.

EXAMPLE 2.12
Given the following polynomials

$$f_1 = x^3 + 2x^2 + 3x + 1$$
$$f_2 = 2x^2 + x + 2$$

the result of f_1 pseudo divided by f_2 according to Definition 2.41 is given by the polynomials $q = 2x+3$ and $r = 5x-2$, since then Equation 2.19 holds: $2^2 \cdot f_1 = (2x+3) \cdot f_2 + (5x-2)$. The same result can be achieved by the following matrix operations:

$$\mathbf{M} = MATRIX((f_1, x \cdot f_2, f_2)) = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 2 & 1 & 2 & 0 \\ 0 & 2 & 1 & 2 \end{bmatrix} \mapsto \cdots \mapsto \begin{bmatrix} 2 & 1 & 2 & 0 \\ 0 & 2 & 1 & 2 \\ 0 & 0 & 5 & -2 \end{bmatrix} = \mathbf{M}'$$

Obviously, the pseudo remainder coefficients can be taken from the last row of $\mathbf{M}'$, where $\mathbf{M}'$ is obtained from $\mathbf{M}$ by elementary row operations.

DEFINITION 2.45
Let $\mathbf{M} \in U^{p \times l}, l \geq p$. *Define* $\mathbf{M^{(i)}} \in U^{p \times p}$ *for* $i = p, \ldots, l$ *as the* $p \times p$ *square submatrix of* $\mathbf{M}$ *consisting of the first* $(p-1)$ *columns of* $\mathbf{M}$ *combined with the* i^{th} *column of* $\mathbf{M}$. *The* ***determinant polynomial*** *of* $\mathbf{M}$ *is then defined as follows:*

$$DetPol(\mathbf{M}) = \sum_{i=p}^{l} det(\mathbf{M^{(i)}}) \cdot x^{l-i}. \tag{2.24}$$

Observe that $DetPol(\mathbf{M}) = 0$ *if* $l < p$. *Remember that* $det(M^{(i)})$ *defines the determinant of the matrix* $M^{(i)}$.

DEFINITION 2.46
*The **sylvestermatrix M** of f_1 and f_2 with $deg(f_1) = m$ and $deg(f_2) = h$ is received by:*

$$\mathbf{M} = MATRIX((x^{h-1}f_1, x^{h-2}f_1, \ldots, f_1, x^{m-1}f_2, x^{m-2}f_2, \ldots, f_2)). \tag{2.25}$$

DEFINITION 2.47
Define $\mathbf{M_i}$ of the sylvestermatrix $\mathbf{M}$ of f_1 and f_2 with $deg(f_1) = m$ and $deg(f_2) = h$ as the matrix obtained from $\mathbf{M}$ by deleting the last i rows belonging to f_1, the last i rows corresponding to f_2 and the last i columns:

$$\mathbf{M_i} = MATRIX((x^{h-i-1}f_1, x^{h-i-2}f_1, \ldots, f_1, x^{m-i-1}f_2, x^{m-i-2}f_2, \ldots, f_2)). \tag{2.26}$$

Then, the matrix $\mathbf{M_i}$ has $m + h - 2i$ rows and $m + h - i$ columns.

DEFINITION 2.48 (Subresultant)
*The **i^{th} subresultant** of f_1 and f_2 with $deg(f_1) = m$ and $deg(f_2) = h$ is then defined using the Definitions 2.45 - 2.47:*

$$SubRes_i(f_1, f_2) = DetPol(\mathbf{M_i}) = det(\mathbf{M_i^{(m+h-2i)}}) \cdot x^i + \cdots + det(\mathbf{M_i^{(m+h-i)}}), \tag{2.27}$$

where $0 \leq i < min(m, h)$.

DEFINITION 2.49 (Principal subresultant coefficient)
*The nominal leading coefficient of $SubRes_i$ is called the **i^{th} principal subresultant coefficient** of f_1 and f_2 with $deg(f_1) = m$ and $deg(f_2) = h$:*

$$psc_i(f_1, f_2) = det(\mathbf{M_i^{(m+h-2i)}}). \tag{2.28}$$

*The **psc set** of f_1 and f_2 is defined as:*

$$PSC(f_1, f_2) = \{ \bigcup_{i=0}^{min(m,h)} psc_i(f_1, f_2) \mid psc_i \neq 0\}. \tag{2.29}$$

DEFINITION 2.50 (Subresultant chain)
*The **subresultant chain** of f_1 and f_2 with $deg(f_1) = m$ and $deg(f_2) = h$ is $(S_j)_{j=0}^{r+1}$, where*

$$\begin{aligned} S_{r+1} &= f_1, \\ S_r &= f_2, \\ S_{r-1} &= SubRes_{r-1}(f_1, f_2), \\ &\vdots \\ S_0 &= SubRes_0(f_1, f_2). \end{aligned} \tag{2.30}$$

*A member $SubRes_i(f_1, f_2)$ in the subresultant chain is **regular** iff its degree is equal to the nominal degree i; otherwise it is **irregular**. The chain is **regular** iff $SubRes_i(f_1, f_2)$ is regular $\forall i = 0, \ldots, r - 1$.*

One can easily imagine how to receive a principle subresultant coefficient chain from a subresultant chain. The following lemma explains the correspondance of the psc chain of two polynomials and the degree of their greatest common divisor.

LEMMA 2.9 *Let $f_1, f_2 \in K[x_1, \ldots, x_{n-1}][x_n]$, where $deg(f_1)_{x_n} = m, deg(f_2)_{x_n} = h$. Then for all $0 < i \leq min(m, h)$ the following two statements are equivalent:*

(i) f_1 and f_2 have a common factor of degree i,

(ii) $psc_j(f_1, f_2) = 0$ for $j = 0, \ldots, i-1$ and $psc_i(f_1, f_2) \neq 0$.

In the following, Sturm Sequences are defined which can be used to calculate the number of distinct real zeros of a polynomial within a certain interval of interest. Even though the theory of Sturm Sequences supersedes Descartes' rule of sign and its generalisations as a tool for root counting, Descartes' rule of sign is presented as well because it is sufficient for various applications, and it is quite straightforward. Descartes' rule of sign can be applied to the simple case that all zeros of the polynomial are real, whereas Sturm Sequences can be applied to polynomials with real and complex zeros. Before defining Descartes' rule of sign, sign variations need to be explained first.

DEFINITION 2.51
*Let $\hat{\alpha} = (\alpha_0, \cdots, \alpha_h)$ be a sequence of real numbers. A **sign variation** in $\hat{\alpha}$ at position i $(i = 1, \cdots, h)$ occurs if for some $j = 0, \cdots, i-1$ the following two conditions are satisfied:*

$$\text{(i)}\ \ \alpha_j \cdot \alpha_i < 0 \quad \text{and}$$

$$\text{(ii)}\ \ \alpha_{j+1} = \alpha_{j+2} = \cdots = \alpha_{i-1} = 0.$$

The number of sign variations in $\hat{\alpha}$ is denoted by $Var(\hat{\alpha})$.

EXAMPLE 2.13
Take the following sequence as an example: $\hat{\alpha} = (0, -1, 0, 3, 8, -7, 9, 0, 0, 8)$. It has sign variations at the positions 3, 5 and 6. The sequence has a sign variation at position 5 ($\alpha_5 = -7$) since condition (i) is satisfied with $\alpha_4 = 8$ and condition (ii) is trivially true because there are not any α's between position 4 and 5 which can be considered. Therefore, the total number of sign variations in $\hat{\alpha}$ is $Var((0, -1, 0, 3, 8, -7, 9, 0, 0, 8)) = 3$.

THEOREM 2.10 (Descartes' Rule of Sign) *Let $f = \sum_{i=0}^{n} a_i x_i$ be a univariate polynomial. Then, the number n_s of sign variations in the sequence $(a_n, a_{n-1}, \cdots, a_1, a_0)$ of coefficients of f is greater than the number n_p of positive real roots of f by some nonnegative even number. If all roots of f are real, then $n_s = n_p$.*

The history of counting real roots started with Descartes' rule of sign which has been established in 1637. At the beginning of the 19 th century, Budan and Fourier discovered the first generalization of Descartes' rule of sign. They (i) defined a sequence D of higher nonzero derivatives of a polynomial f, (ii) denoted the number of real roots of f in the interval (a, b) by n, and (iii) introduced s as the difference of the sign variations in the sequences $D(a)$ and $D(b)$. Then, they could prove that $n \leq s$, $s - n$ is even, and $n = s$ iff all roots of f were real. Only some years later, Sturm replaced the sequence D by his Sturm Sequence, which is obtained from f and its first derivative f' by successive polynomial division with negative remainder, in order determine the exact number n of real roots of f in the interval (a, b). Now, s denotes the difference of the sign variations in the sequences $S(a)$ and $S(b)$. Using Sturm Sequences, the number n of real roots of f in the interval (a, b) equals to s. In 1853, Sylvester extended Sturm's root counting

method with an additional polynomial side condition g. The sequence S was changed to T which can be computed from f and $f'g$ by successive polynomial division with negative remainder. Now, s represents the difference of the sign variations in the sequences $T(a)$ and $T(b)$. If n_p and n_n defines the number of positve and negative real roots of f in the interval (a, b), respectively, then $s = n_p - n_n$. A combinatorial extension to cover finitely many side conditions has then been developed by Tarski which leads to the first quantifier elimination method [Tar51].

In the following, Sturm's theory is presented in greater detail. The Sturm Sequence is based on successive polynomial division with negative remainder, i.e. it can be constructed from the polynomial remainder sequence (PRS) with the only exception that the negative remainder is taken instead of the positive one.

DEFINITION 2.52
*A **Sturm Sequence** or **canonical Sturm Sequence** of two polynomials f and g in $K[x]$, denoted as $STURM(f, g)$, is defined as the sequence of polynomials $(h_1, h_2, \cdots, h_p)$, which is constructed as follows:*

$$\begin{aligned}
h_1 &= f \\
h_2 &= g \\
h_1 &= q_1 \cdot h_2 - h_3, && deg(h_3) < deg(h_2) \\
h_2 &= q_2 \cdot h_3 - h_4, && deg(h_4) < deg(h_3) \\
&\vdots \\
h_{p-2} &= q_{p-2} \cdot h_{p-1} - h_p, && deg(h_p) < deg(h_{p-1}) \\
h_{p-1} &= q_{p-1} \cdot h_p + 0, && (2.31)
\end{aligned}$$

*where $h_p = -GCD(h_1, h_2)$, $(q_i, r_i) = PSEUDO\text{-}DIVISON(h_i, h_{i+1})$, and $h_{i+2} = -r_i$. The **standard Sturm Sequence** is given by $STURM(f, f')$, where f' represents the first derivative of f. The **Sturm-Sylvester Sequence** is then represented by $STURM(f, f'g)$.*

In order to illustrate the **Sturm-Sylvester Theorem** to count the number of real zeros with one polynomial side condition g, an important lemma has to be mentioned first. In general, a Sturm Sequence $(h_1, h_2, \cdots, h_p)$ can be applied to an element $a \in K$ by defining $(h_1, h_2, \cdots, h_p)(a) = (h_1(a), h_2(a), \cdots, h_p(a))$.

LEMMA 2.11 *Let f and $g \in K[x]$ with K being a real closed field. Consider the Sturm-Sylvester Sequence of the polynomials f and $f'g$: $STURM(f, f'g) = (h_1, h_2, \cdots, h_p)$. Let (a, b) to be the interval $(a < b)$ containing exactly one zero $c \in (a, b)$ of f. Then the following holds:*

$$VarDiff\left[STURM(f, f'g)\right]_b^a = sign(g(c)),\ where \tag{2.32}$$

$VarDiff\left[STURM(f, f'g)\right]_b^a := Var((h_1, h_2, \cdots, h_p)(a)) - Var((h_1, h_2, \cdots, h_p)(b))$.

A detailed poof of Lemma 2.11 can be found in [Mis93]. Suppose the interval (a, b) contains more than one zero, then $VarDiff\left[STURM(f, f'g)\right]_b^a$ equals just the sum of signs of $g(c)$ for all $c \in (a, b)$ with $f(c) = 0$:

$$VarDiff\left[STURM(f, f'g)\right]_b^a = \sum_{c \in (a,b), f(c)=0} sign(g(c)). \tag{2.33}$$

THEOREM 2.12 (Sturm-Sylvester Theorem) *Let f and g be two polynomials with coefficients in a real closed field K. Suppose $STURM(f, f'g) = (h_1 = f, h_2 = f'g, \cdots, h_p)$. Then for any interval [a,b] $\subseteq K$ $(a < b)$:*

$$c_f\,[g > 0]_b^a - c_f\,[g < 0]_b^a = VarDiff\left[STURM(f, f'g)\right]_b^a \tag{2.34}$$

where $c_f\,[E]_b^a$ denotes the number of distinct real roots of the polynomial f in the interval $[a, b] \subseteq K$ at which the polynomial equation or inequation E holds. Here, the value of a may be $-\infty$, and the value of b may be ∞.

Corollary 2.13 describes how to compute the number of distinct real zeros of a polynomial f in the interval (a, b) when no side condition is applied. It can be easily deduced from Theorem 2.12 by taking g as the constant polynomial 1.

COROLLARY 2.13 *Let $f(x) \in K[x]$ and let $STURM(f, f') = (h_1, h_2, \cdots, h_p)$ be the standard Sturm Sequence as defined in Definition 2.52. Then the number of distinct real roots of f in the interval (a, b) can be computed with:*

$$VarDiff\left[STURM(f, f'g)\right]_b^a \tag{2.35}$$

Corollary 2.14 provides a method to obtain the number of distinct real roots of a polynomial f in the interval (a, b), where g is greater than zero, equal to zero, and less than zero. Thus, the Sturm theory presents a method to solve a polynomial system for two univariate polynomial equations and inequations with the constraint that at least one polynomial equation is available. (This method can be extended to count the number of zeros of f with finitely many side conditions which is not discussed here.)

COROLLARY 2.14 *Let f and $g \in K[x]$ with K being a real closed field. For any interval $(a, b) \subseteq K$ $(a < b)$, one has:*

$$c_f\,[g < 0]_b^a + c_f\,[g = 0]_b^a + c_f\,[g > 0]_b^a = VarDiff\left[STURM(f, f')\right]_b^a \tag{2.36}$$

$$-c_f\,[g < 0]_b^a + c_f\,[g > 0]_b^a = VarDiff\left[STURM(f, f'g)\right]_b^a \tag{2.37}$$

$$c_f\,[g < 0]_b^a + c_f\,[g > 0]_b^a = VarDiff\left[STURM(f, f'g^2)\right]_b^a. \tag{2.38}$$

or expressed with matrix notation:

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} c_f\,[g < 0]_b^a \\ c_f\,[g = 0]_b^a \\ c_f\,[g > 0]_b^a \end{bmatrix} = \begin{bmatrix} VarDiff\,[STURM(f, f')]_b^a \\ VarDiff\,[STURM(f, f'g)]_b^a \\ VarDiff\left[STURM(f, f'g^2)\right]_b^a \end{bmatrix}. \tag{2.39}$$

All the above theorems, lemmas and corollaries of the Sturm theory assume that the considered real roots lie within the open interval (a, b). The question is what happens, if the polynomial vanishes on one or both interval bounds. (The REAL-ROOT-ISOLATION Algorithm of Section 4.2 can be implemented in a simpler way if one knows whether the interval in which the real roots are counted is open, closed or half-open.) The next example investigates this question by counting the number of distinct real roots using Corollary 2.13.

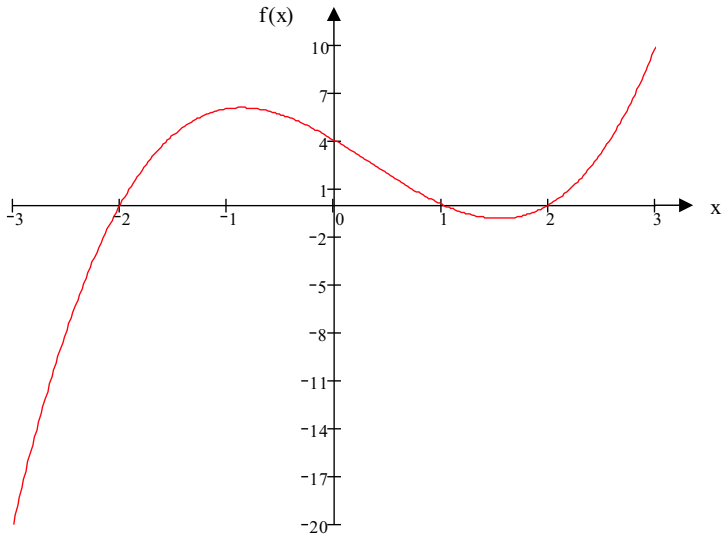

Figure 2.1: 2D-Graph of the polynomial function f(x).

EXAMPLE 2.14
Given the following real polynomial:

$$f = (x-2)(x+2)(x-1) = x^3 - x^2 - 4x + 4$$

and the graphical illustration of the corresponding polynomial function $f(x)$ in Figure 2.1. Then the standard Sturm Sequence of f and f' is as follows:

$$\begin{aligned} h_1 &= f = x^3 - x^2 - 4x + 4 \\ h_2 &= f' = 3x^2 - 2x - 4 \\ h_3 &= \frac{26}{9}x - \frac{32}{9} \\ h_4 &= \frac{324}{169}. \end{aligned}$$

The number of distinct roots for an interval (a, b) can then be calculated according to Corollary 2.13. Table 2.1 presents the results for several chosen intervals. One can observe that the number of distinct roots is counted within the interval $(a, b]$. If a or b happens to be a root of the polynomial f, this means that a would not be counted while b would be counted. A mathematical proof in literature could not be found and therefore this is only an empirical result which has to be pooved in order to become applicable.

Interval (a,b)	$VarDiff\,[STURM(f,f')]_b^a$	♯ of roots	roots
$(-3,3)$	$Var_{-3}(-,+,-,+) - Var_3(+,+,+,+)$	3	$x=-2, x=1, x=2$
$(-2,3)$	$Var_{-2}(0,+,-,+) - Var_3(+,+,+,+)$	2	$x=1, x=2$
$(-2,1)$	$Var_{-2}(0,+,-,+) - Var_1(0,-,-,+)$	1	$x=1$
$(0,1)$	$Var_0(+,-,-,+) - Var_1(0,-,-,+)$	1	$x=1$
$(1,2)$	$Var_1(0,-,-,+) - Var_2(0,+,+,+)$	1	$x=2$
$(1,3)$	$Var_1(0,-,-,+) - Var_3(+,+,+,+)$	1	$x=2$
$(2,3)$	$Var_2(0,+,+,+) - Var_3(+,+,+,+)$	0	

Table 2.1: Number of distinct roots in the interval (a,b).

2.4 Ideal Theory

This section introduces the concept of Ideals.

DEFINITION 2.53
*A subset $J \subseteq R$ of a ring R is an **ideal** if it satisfies the following properties:*

$$1) \quad a,b \in J \Rightarrow a-b \in J \tag{2.40}$$
$$2) \quad c \in R,\ a \in J \Rightarrow ca \in J \tag{2.41}$$

The first condition states that J is an additive subgroup of the additive group of R and thus the zero element 0 belongs to J. The second condition implies that J is closed under multiplication with ring elements.

DEFINITION 2.54
*An ideal J is said to be **radical** iff for all integers $s \geq 1$, $f^s \in J$ implies $f \in J$. The ideals $\{0\}$ and R are called **improper** ideals of R while all other ideals are **proper**. A subset $F \subseteq J$ that generates J is called a **basis** or a **system of generators** of the ideal J. The ideal consisting of all polynomials that vanish on the same zero set Z is denoted by $Ideal(ZERO(Z))$.*

The construction of an ideal is illustrated by the following theorem.

THEOREM 2.15 *Let $a_1, \cdots, a_m \in R$ and $R \subseteq R'$. The ideal generated in R' by the basis $a_1, \cdots, a_m$ is denoted by $Ideal_{R'}((a_1, \cdots, a_m))$ and is constructed as follows:*

$$Ideal_{R'}(a_1, \cdots, a_m) = \{\sum_{i=1}^{m} a_i \cdot b_i \mid b_i \in R'\}. \tag{2.42}$$

In obvious situations, the subscript R' can be omitted.

DEFINITION 2.55
*An ideal is called a **principle ideal** if the ideal is generated by a single element. A **principle ideal ring** or **principle ideal domain**, for short PID, is one in which every ideal is principle.*

The principle ideal generated by the single basis element 0 is $Ideal(0) = \{0\}$, and the one constructed by 1 is $Ideal(1) = R$. Those two principle ideals are the improper ideals of the ring R. For example, $R[x]$, the univariate polynomial ring, is a principle ideal domain because the ideal generated by a set of polynomials F can also be generated by a single generator which is the greatest common divisor (GCD) of all polynomials in F (see Algorithm 2.1). Furthermore, the single generator of every ideal in $R[x]$ is unique up to multiplication by a nonzero constant in R.

DEFINITION 2.56
Let $F \subseteq R[x_1, \cdots, x_n]$ be a set of polynomials. Then, the ***head ideal*** *$Head(F)$ of the set F is defined to be the ideal generated by $hterm(F)$: $Head(F) = Ideal[hterm(F)]$.*

DEFINITION 2.57
Let $F \subseteq R[x_1, \cdots, x_n]$ be a set of polynomials and let $ZERO(F)$ be the zero set of these polynomials. Then, $Ideal(ZERO(F))$ defines the ideal that contains all polynomials whose corresponding polynomial equations vanish on the same set of zeros. The ideal is said to be generated by the zero set $ZERO(F)$.

DEFINITION 2.58
A ring R is called ***Noetherian*** *if any ideal of R has a finite system of generators.*

The Hilbert Basis Theorem extends this definition. It says that, if R is Noetherian, the polynomial ring $R[x_1, \cdots, x_n]$ is also Noetherian.

The next definition gives the five basic operations defined on ideals.

DEFINITION 2.59
Let $I, J \subseteq R$ be two ideals. Then, the basic ideal operations are:

(1) ***Sum****: $I + J$ is the smallest ideal containing both I and J.*
$I + J = \{a + b \mid a \in I \text{ and } b \in J\}$.

(2) ***Product****: IJ is the ideal generated by $\{ab \mid a \in I \text{ and } b \in J\}$.*

(3) ***Intersection****: $I \cap J$ is the set-theoretic intersection of I and J.*
$I \cap J = \{a \mid a \in I \text{ and } a \in J\}$.

(4) ***Quotient****: $I : J$ is defined as the set $\{a \in R \mid aJ \subseteq I\}$.*

(5) ***Radical(J)****: $\sqrt{J}$ is defined to be the set $\{a \in R \mid n \in \mathbb{N}, \exists n \geq 1 a^n \in J\}$.*

DEFINITION 2.60
A ***quotient ring****, also called a* ***residue-class ring****, is a ring that is the quotient of a ring R and one of its ideals I, denoted by R/I. Basically, a quotient ring defines a set of equivalence classes or residue classes. Let $[x]$ and $[y]$ be two equivalence classes of the quotient ring R/I. Then, it holds that $[x] = [y]$ iff $x - y \in I$.*

Chapter 3

Solving the Decision Problem using Gröbner Bases

In this chapter, Buchberger's algorithm for constructing Gröbner bases is analyzed to solve the decision problem for a system of non-linear multivariate polynomial equations and inequations with respect to the application background of automated reasoning with description logics. Buchberger introduced the concept of Gröbner bases as bases for ideals in polynomial rings over a field first in 1965 (Thesis, Innsbruck). Buchberger named them in honor of his thesis advisor W. Gröbner (1899-1980). This chapter discusses Buchberger's Algorithm as well as several methods to solve real polynomial systems using Gröbner bases. It concludes with assessments of the presented Gröbner bases methods.

3.1 Buchberger's Algorithm

Gröbner bases are special generators for ideals in polynomial rings over a field. They offer solutions for a variety of algebraic problems and present more convenient computations to many problems in polynomial algebra. Basically, Buchberger's algorithm can be described as a combination of the Euclidean algorithm for two univariate polynomials and the Gaussian elimination for a system of linear multivariate polynomials, (see also [Yap00]). Buchberger's algorithm is applicable for the non-linear multivariate case. Thus, the Euclidean algorithm and the Gaussian elimination are considered first. A relationship between these two and Buchberger's algorithm is established later on.

First, an introductory example illustrates how a system of polynomial equations is solved with mathematical tools from school. This is then extended to the Gaussian elimination method which applies matrix computations.

EXAMPLE 3.1
Assume the following real linear polynomial system:

$$\begin{aligned} &(1) \quad -x_1 - 2x_2 + 5x_3 = 2 \\ &(2) \quad 3x_1 + x_2 = 9 \\ &(3) \quad 2x_1 - 4x_2 + 3x_3 = -2. \end{aligned} \tag{3.1}$$

Using the method from school to solve this polynomial system two different pairs of polynomial equations have to be combined to obtain two new polynomial equations such that

the same variable has been eliminated. This means, two polynomials are added such that the monomials of the same variable vanish. Eliminating, for example, x_1, one yields:

$$\begin{aligned} (A) \quad & -8x_2 + 13x_3 = 2 \\ (B) \quad & -5x_2 + 15x_3 = 15 \end{aligned}$$

Equation (A) is received by multiplying equation (1) by 2 and adding it to equation (3). Similarily, equation (B) results from the multiplication from equation (1) with 3 and adding it to equation (2). Then, equation (A) and (B) are multiplied by -5 and 8, respectively, and the results are added to obtain the final polynomial equation in one variable:

$$55x_3 = 110.$$

This equation can be solved for x_3 which gives $x_3 = 2$. By inserting this result in equation (A) or (B) one obtains $x_2 = 3$. Finally, x_1 can be computed to be equal to $x_1 = 2$ by replacing the values for x_2 and x_3 in equation (1) or (3). Thus, the solution of the real linear polynomial system in (3.1) is $(2, 3, 2) \in \mathbb{R}^3$.

For computers, it is more convenient to solve the same task using the Gaussian elimination because Gaussian elimination is based on matrix computations. First, the coefficients of the real linear polynomial system are inserted into a $m \times n$ matrix $\mathbf{A}'$. Remember that a $m \times n$ matrix consists of m rows and n columns. Therefore, each polynomial is traditionally written in the form such that the monomial containing an arbitrary coefficient and the power product 1, $(1 = \mathbf{x^0})$, is on the right hand side of the equal sign whereas all other monomials are on the left hand side. The polynomial system in Equation (3.1) is already in this form. The coefficients with their signs are inserted into the matrix just by dropping the equal signs and the letters for the variables. Each equation corresponds to one row in matrix $\mathbf{A}'$ such that column n contains the coefficients of the monomials with the power product 1 and column i, for $1 \leq i \leq n-1$, contains the coefficients of the monomials where the power product is equal to a single variable x_i. (Note that linear polynomials consist only of such monomials.) This matrix is transformed to **row echelon** form by applying elementary row operations (or elementary column operations but not both). A $m \times n$ matrix is in row echelon form if, for all $1 \leq i \leq m$, the first non-zero entry of row $i+1$ is to the right of the first non-zero entry of row i. (This is a staircase form.) The result is a reduced polynomial system which can be solved easily.

EXAMPLE 3.2 (Gaussian elimination)
Now, the Gaussian elimination method is used to solve the system of polynomials in (3.1). The matrix of coefficients is then given by the first of the following three matrices which is transformed to row echelon form by elementary row operations:

$$\mathbf{A}' = \begin{bmatrix} -1 & -2 & 5 & 2 \\ 3 & 1 & 0 & 9 \\ 2 & -4 & 3 & -2 \end{bmatrix} \longmapsto \begin{bmatrix} -1 & -2 & 5 & 2 \\ 0 & -5 & 15 & 15 \\ 0 & -8 & 13 & 2 \end{bmatrix} \longmapsto \begin{bmatrix} -1 & -2 & 5 & 2 \\ 0 & -5 & 15 & 15 \\ 0 & 0 & -11 & -22 \end{bmatrix}.$$

The second matrix can be obtained from the first one by multiplying the first row with 3 and adding the result to the second row and by multiplying the first row with 2 and adding the result to the third row. The third matrix is computed from the second one

by multiplying the second row with $-8/5$ and adding the result to the third row. The reduced real linear polynomial system can be extracted from the last matrix:

$$\begin{aligned} (1') \quad & -x_1 - 2x_2 + 5x_3 = 2 \\ (2') \quad & -5x_2 + 15x_3 = 15 \\ (3') \quad & -11x_3 = -22. \end{aligned}$$

Again, x_3 can be received from equation $(3')$, x_2 is obtained from equation $(2')$ by inserting the result of x_3 and x_1 can then be calculated from equation $(1')$. This also yields $(2, 3, 2)$.

Note that the set of solutions is not altered when the matrix $\mathbf{A}'$ is transformed to row echelon form using elementary operations. The dimension of the solution space X for a polynomial system can be obtained from a matrix $\mathbf{A}$ by counting its number of columns and subtracting its rank: $dim(X) = n - rank(\mathbf{A})$. The rank of a matrix is equal to the rank of columns which in turn is equal to the rank of rows for all $m \times n$ matrices. The rank of columns or the rank of rows of a matrix can be determined by the number of linear independent columns or rows, respectively. Matrix $\mathbf{A}$ can be obtained from the polynomial system when it is written in matrix notation as $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is the vector of the variables involved, $\mathbf{b}$ is the vector of the coefficients of the monomials containing the power product 1, and $\mathbf{A}$ is the matrix of the coefficients of all other monomials. Extending matrix $\mathbf{A}$ with vector $\mathbf{b}$ as an additional column, one yields the matrix $\mathbf{A}'$. For the above example, the rank of $\mathbf{A}$ is equal to 3 as well as the number of columns in $\mathbf{A}$. Therefore, the dimension of the solution space is 0.

The Gaussian elimination method is also applicable if the dimension of the solution space is greater than 1. In this case, there exists at least one column where no edge of the steps in the row echelon form of matrix $\mathbf{A}'$ occurs. The variable that corresponds to such a column is a free parameter. The solution space is not empty if $rank(\mathbf{A}) = rank(\mathbf{A}')$. The polynomial system is said to be **universely solvable**, i.e. it is solvable for each chosen $\mathbf{b}$, if $rank(\mathbf{A}) = m$. Also, the polynomial system is said to be **uniquely solvable**, i.e. it has exactly one solution, if $rank(\mathbf{A}) = rank(\mathbf{A}') = n$.

Now, the Euclidean algorithm is discussed with respect to the task of solving a system of real non-linear univariate polynomial equations $f_1 = f_2 = \cdots = f_p = 0$. In this case, solving means to compute the zero set of the polynomials: $ZERO(f_1, \cdots, f_p)$. Corollary 3.1 provides the basis for this.

COROLLARY 3.1 *The zero set of a set of polynomials $f_1, \cdots, f_p$ is equivalent to the zero set of the ideal J generated by the same set of polynomials ($J = Ideal(f_1, \cdots, f_p)$):*

$$ZERO(f_1, \cdots, f_p) = ZERO(J). \tag{3.2}$$

For univariate polynomials, $K[x]$ is a principle ideal domain, where K is a field. Remember that in a principle ideal domain, every ideal can be generated by a single polynomial. Therefore, in order to solve the system of univariate polynomials, one can simply compute this single generator of the ideal of the polynomials $f_1, \cdots, f_p$ and calculate the zeros of this single equation. These zeros represent the common zeros of all polynomials. The following corollary explains the relationship between ideals in $K[x]$ and the greatest common divisor.

COROLLARY 3.2 *Let $f_1, \cdots, f_p \in K[x]$ be polynomials, where K is an arbitrary field. Then, $gcd(f_1, \cdots, f_p)$ is a generator of the ideal $Ideal(f_1, \cdots, f_p)$.*

Thus, the single generator of the ideal of the polynomials $f_1, \cdots, f_p$ is the greatest common divisor of $f_1, \cdots, f_p$. The greatest common divisor for two univariate polynomials can be computed using the Euclidean algorithm as presented in Section 2.2 (Algorithm 2.1). Definition 2.32 states that the greatest common divisor of $p \geq 3$ polynomials can be determined by the following recursion: $gcd(f_1, \cdots, f_p) = gcd(f_1, gcd(f_2, \cdots, f_p))$.

EXAMPLE 3.3
Assume the following real non-linear univariate polynomial system:

$$\begin{aligned} f_1 : \quad & x^3 - 3x + 2 = 0 \\ f_2 : \quad & x^4 - 1 = 0 \\ f_3 : \quad & 2x^3 + 2x^2 + 2x + 2 = 0. \end{aligned} \tag{3.3}$$

The common solutions of the three equations are presented by the zero set $ZERO(f_1, f_2, f_3)$, which is equal to the zero set of the single generator of the ideal $Ideal(f_1, f_2, f_3)$. The single generator is $gcd(f_1, f_2, f_3)$, which is computed as follows:

$$\begin{aligned} gcd(f_1, f_2, f_3) &= gcd(x^3 - 3x + 2, x^4 - 1, 2x^3 + 2x^2 + 2x + 2) \\ &= gcd(x^3 - 3x + 2, gcd(x^4 - 1, 2x^3 + 2x^2 + 2x + 2)) \\ &= gcd(x^3 - 3x + 2, x^2 - 1) \\ &= x - 1. \end{aligned}$$

Thus, $Ideal(f_1, f_2, f_3) = Ideal(x - 1)$. The zero of this single generator can be easily computed to be $x = 1$, which is also the common zero of the set of polynomials in (3.3).

Two special cases have been considered: a multivariate but linear polynomial system and a univariate but non-linear polynomial system. The decision problem can be solved for both cases. Unfortunately, it is rather complicated to solve the decision problem for a non-linear multivariate polynomial system with respect to real solutions. One possibility is to apply solution techniques using Gröbner bases. Therefore, Gröbner bases are introduced and discussed here, and related solution methods for the decision problem are presented and analyzed.

The basic concept in Gröbner bases is that of reduction.

DEFINITION 3.1
Let f, g be two polynomials in $R[\mathbf{x}]$. Then, f is ***reducible*** *by g if $hmono(g)$ divides some monomial p in f, i.e. $p = q \cdot hmono(g)$ for some monomial q. This is denoted by $f \xrightarrow{g} h$, where $h = f - q \cdot g$ represents the* ***reduct*** *of f by g. The polynomial f is reducible by a set of polynomials G, if there exists a g in G such that $f \xrightarrow{g} h$. If f is not reducible by G, f is called to be in* ***G-normal form****, written as $f \xrightarrow{G}$. A* ***normal form*** *of f is then any G-normal form $\hat{f}$ such that $f \xrightarrow[*]{G} \hat{f} \xrightarrow{G}$, where $\xrightarrow[*]{G}$ denotes the reflexive transitive closure of $\xrightarrow{G}$. $NF_G(f)$ indicates the set of all G-normal forms of f.*

Algorithm 3.1 computes a normal form for a polynomial $f \in R[\mathbf{x}]$ and a finite set $G \subseteq R[\mathbf{x}]$. The basic idea is to cancel the head monomial in f with respect to a fixed monomial ordering by multiplying some g_i by an appropriate monomial and subtracting the result. The algorithm is based on the fact that every $f \in K[x_1, \cdots, x_n]$ can be written with respect to an ordered p-tuple of nonzero polynomials $F = (f_1, \cdots, f_p)$ and a fixed admissible ordering as:

$$f = a_1 f_1 + \cdots + a_p f_p + r, \tag{3.4}$$

where $a_i, r \in K[x_1, \cdots, x_n]$. Then, either $r = 0$ or r is a polynomial containing only monomials, none of which can be divided by the head monomial of F, i.e. by the set of head monomials of any polynomial in F: $hmono(f_1), \cdots, hmono(f_p)$. Also, if $a_i f_i \neq 0$, then $multideg(f) \geq multideg(a_i f_i)$. r is called a remainder of f on division by F. Note that Equation (3.4) is the multivariate equivalent to the univariate case in Equation (2.10). The NORMAL-FORM algorithm is basically the multivariate version of the DIVISON algorithm, Algorithm 2.2, used in the GCD algorithm. The NORMAL-FORM algorithm computes the remainder f on division by F. Note that the pseudo division as presented by in Section 2.3 cannot be applied here.

Algorithm 3.1 NORMAL-FORM(f, G)

Input: A polynomial $f \in R[\mathbf{x}]$ and a finite set $G \subseteq R[\mathbf{x}]$.
Output: Normal form of the polynomial f.

```
p := f; h := 0; s := |G|;
G-tuple := N-TUPLE(G);
while p ≠ 0 do
  i := 1;
  division_occurred := FALSE;
  while (i ≤ s) and (¬division_occurred) do
    if HMONO(g_i) divides HMONO(p) then
      p := p − (HMONO(p)/HMONO(g_i)) · g_i;
      division_occurred := TRUE;
    else
      i := i + 1;
    end if
  end while
  if NOTdivision_occurred then
    h := h + HMONO(p);
    p := p − HMONO(p);
  end if
end while
return h ;
```

The N-TUPLE algorithm called in the NORMAL-FORM algorithm takes an arbitrary set and returns it as an ordered n-tuple. The monomial $hmono(g)$ *divides* the monomial $hmono(f)$ iff $hmomo(f)$ can be expressed as in the univariate case according to Equation (2.10) as: $hmomo(f) = q \cdot hmono(g) + r$ and where $r = 0$. In case $r = 0$, the devision operator / can be applied, and $q = hmono(f)/hmono(g)$ is explicitly calculated. The NORMAL-FORM algorithm terminates because the $multideg(f)$ gets smaller at each

step of the loop by one of the following two actions: If $hmono(g_i)$ *divides* $hmono(f)$, then f is reduced of its head monomial by multiplying some g_i by an appropriate monomial and subtracting the result from f. If no $hmono(g_i)$ *divides* $hmono(f)$, the head monomial of f is removed from f and added to the remainder. In both cases, $multideg(f)$ is reduced, and at some point f becomes equal to zero. When the algorithm terminates, r is of the required form since a monomial is only added to r if it is not divisible by any of the head monomials of the polynomials in G.

Note that a G-normal form of f as defined in Definition 3.1 and as implemented in the NORMAL-FORM algorithm is not always unique. The G-normal form can depend on the selection of g_i in case more than one head monomial divides the head monomial of the input polynomial f. In the NORMAL-FORM algorithm, a sequential selection is implemented since the set G is transformed to an ordered n-tuple. Therefore, the ordering of the input set G might produce different outputs.

EXAMPLE 3.4
This example illustrates the computation of the normal form of $f = 2y^4 + x^3y - 3xy + 1$ with respect to the set $G = \{x^2y, y^2\}$ using Algorithm 3.1. First, let p be equal to f, h to 0 and s to 2.

1) $i := 1$ and $division\_occurred := false$

- $i = 1$: x^2y does not divide $2y^4 \longrightarrow i := 2$
- $i = 2$: y^2 divides $2y^4 \longrightarrow p := (2y^4 + x^3y - 3xy + 1) - \frac{2y^4}{y^2} \cdot y^2 = x^3y - 3xy + 1$ $\longrightarrow division\_occurred := true$

2) $i := 1$ and $division\_occurred := false$

- $i = 1$: x^2y divides $x^3y \longrightarrow p := (x^3y - 3xy + 1) - \frac{x^3y}{x^2y} \cdot x^2y = -3xy + 1$ $\longrightarrow division\_occurred := true$

3) $i := 1$ and $division\_occurred := false$

- $i = 1$: x^2y does not divide $-3xy \longrightarrow i := 2$
- $i = 2$: y^2 does not divide $-3xy$

$h := -3xy$ and $p := 1$

4) $i := 1$ and $division\_occurred := false$

- $i = 1$: x^2y does not divide $1 \longrightarrow i := 2$
- $i = 2$: y^2 does not divide 1

$h := -3xy + 1$ and $p := 0$

5) $h = -3xy + 1$ is returned

There are several characterizations of Gröbner bases. All of them are useful for the general understanding, but only Buchberger's characterization directly implies a construction for a Gröbner basis. The characterizations of Gröbner bases and are presented in the following relative to some fixed admissible ordering $\underset{A}{\leq}$.

DEFINITION 3.2 (Normal Form Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is ***Gröbner*** *iff every $f \in J$ has a unique G-normal form as defined in Definition 3.1. G is said to be a* ***Gröbner basis*** *for the ideal J and $J = Ideal(G)$.*

Suppose there is a set $F = \{f_1, \cdots, f_p\}$ of polynomials which generate the ideal $Ideal(F)$. Let $G = \{g_1, \cdots, g_q\}$ be a Gröbner basis for $Ideal(F)$, where the g_j's are different from the f_i's. Then, $Ideal(F) = Ideal(G)$ and for all $f \in Ideal(F)$ it holds that $|NF_G(f)| = 1$ since each $f \in Ideal(F)$ has a unique G-normal form per definition.

DEFINITION 3.3 (Standard Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is called a ***Gröbner basis*** *of the ideal J iff $Head(G) = Head(J) = Head[Ideal(G)]$.*

This means that a set $G = \{g_1, \cdots, g_q\} \subset J$ is a Gröbner basis of J iff the head monomial of any element of J is divisible by one of the head monomials of the g_i's.

EXAMPLE 3.5
Let $J = Ideal(f_1, f_2)$, where $f_1 = x^3 - 2xy$ and $f_2 = x^2y - 2y^2 + x$ are polynomials in $K[x, y]$. $G = \{f_1, f_2\}$ is not a Gröbner basis of the ideal J. This can be seen as follows: Using total degree ordering, one can show, that x^2 is a member of the ideal J by computing: $x \cdot (x^2y - 2y^2 + x) - y \cdot (x^3 - 2xy) = x^2$. Therefore, $x^2 = hmono(x^2) \in Head(J)$. But $x^2 \notin Head(G)$. The set G is not a Gröbner basis since x^2 is not divisible by $hmono(f_1) = x^3$ or $hmono(f_2) = x^2y$. It follows that $Head(G) \subset Head(J)$ but not $Head(J) = Head(G)$ as required by Definition (3.3) and therefore G is not a Gröbner basis of J, either.

The next characterization is the basis for Buchberger's algorithm of constructing Gröbner bases. First, the definition of the least common multiple of two power products is presented.

DEFINITION 3.4
Let $p, q \in PP$ be two power products, written as $p = x_1^{e_1}x_2^{e_2}\cdots x_n^{e_n}$ and $q = x_1^{d_1}x_2^{d_2}\cdots x_n^{d_n}$. Then, the ***least common multiple (LCM)*** *of p and q is defined as*

$$LCM(p, q) = x_1^{max(e_1,d_1)}x_2^{max(e_2,d_2)}\cdots x_n^{max(e_n,d_n)}. \tag{3.5}$$

The least common multiple is extended for monomials $a \cdot p$ and $b \cdot q$ such that $LCM(a \cdot p, b \cdot q) = a \cdot b \cdot LCM(p, q)$.

EXAMPLE 3.6
Given the following two monomials: $p = 6x^3y$ and $q = 4x^2y^2$. Then, the least common multiple is computed to be $LCM(p, q) = 24x^3y^2$.

DEFINITION 3.5 (Buchberger's Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is a ***Gröbner basis*** *of the ideal J iff the S-polynomial $S(f, g) \xrightarrow[*]{G} 0$ for all $f, g \in G$, where m is given by $m = LCM[hterm(f), hterm(g)]$ and the S-polynomial is defined as follows:*

$$S(f, g) = \frac{m}{hmono(f)}f - \frac{m}{hmono(g)}g. \tag{3.6}$$

EXAMPLE 3.7
Let $f = xy - 2$ and $g = 3x^2 - 1$ be two polynomials. Then, the least common multiple of $hterm(f)$ and $hterm(g)$ is equal to $m = LCM(xy, x^2) = x^2y$ and the S-polynomial can be computed to be:

$$S(f,g) = \frac{x^2y}{xy}f - \frac{x^2y}{3x^2}g = x(xy-2) - \frac{y}{3}(3x^2-1) = -2x + \frac{1}{3}y$$

As illustrated in the example for the standard characterization of Gröbner bases, Example 3.5, a set $G = \{g_1, \cdots, g_q\}$ fails to be a Gröbner basis if a suitable combination of the g_i's create a polynomial whose head monomial is not in the ideal generated by $Head(G)$. This means that the created head monomial is not divisible by any head monomial of G. This situation can occur if the head monomials in the suitable combination just cancel each other. Therefore, an S-polynomial $S(g_i, g_j)$ is designed such that the head monomials of g_i and g_j cancel. If all S-polynomials for all $g_i, g_j \in G$ can be reduced to 0 by the set G, i.e. the head monomial of $S(g_i, g_j)$ is divisible by some head monomial of G, G is said to be a Gröbner basis according to Definition 3.5.

The subsequent characterization relates Gröbner bases with the Church-Rosser property. Consider $\underset{*}{\longrightarrow}$ in the following as a partial order on an arbitrary set U.

DEFINITION 3.6
A partial order $\underset{*}{\longrightarrow}$ is said to be ***Noetherian*** *if it has no infinite descending sequences in the sense of $f_1 \underset{*}{\longrightarrow} f_2 \underset{*}{\longrightarrow} f_3 \underset{*}{\longrightarrow} \cdots$.*

DEFINITION 3.7
A partial order $\underset{*}{\longrightarrow}$ is ***Church-Rosser*** *or* ***confluent*** *if for all $f, g, g' \in U$ if $f \underset{*}{\longrightarrow} g$ and $f \underset{*}{\longrightarrow} g'$ then there exists an $h \in U$ such that $g \underset{*}{\longrightarrow} h$ and $g' \underset{*}{\longrightarrow} h$. The element h is then called a* ***common successor*** *of g and g'. A minimal element k in this partial order is a* ***normal form*** *of g if $g \underset{*}{\longrightarrow} k$.*

THEOREM 3.3 *If $\underset{*}{\longrightarrow}$ is a Noetherian partial order on U then $\underset{*}{\longrightarrow}$ is Church-Rosser iff every $g \in U$ has a unique normal form.*

DEFINITION 3.8 (Church-Rosser Property Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is a ***Gröbner basis*** *of the ideal J iff the relation $\underset{*}{\overset{G}{\longrightarrow}}$ is Church-Rosser.*

The extended standard characterization can be easily derived from the construction of an ideal, as described in Theorem 2.15.

DEFINITION 3.9 (Extended Standard Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is a ***Gröbner basis*** *of the ideal J iff for all $f \in J$, there are elements $\alpha_i \in R[\mathbf{x}]$, $g_i \in G$, $i = 1 \cdots m$ (i.e. $m = |G|$) such that*

$$f = \sum_{i=1}^{m} \alpha_i g_i,$$

and $hterm(f) \underset{A}{\geq} hterm(\alpha_i g_i)$ for all i.

DEFINITION 3.10 (Ideal Membership Characterization)
A finite subset G of an ideal $J \subseteq R[\mathbf{x}]$ in $R[\mathbf{x}]$ is a ***Gröbner basis*** *of the ideal J iff $\forall f \in J$, $f \xrightarrow[*]{G} 0$. Clearly, $J = Ideal(G)$ then.*

Suppose that h is the G-normal form of f. Then, $h \equiv f[modIdeal(G)]$. If G is Church-Rosser, then each equivalence class of polynomials modulo $Ideal(G)$ has a unique representative. Obviously, the unique representative of $Ideal(G)$ is the zero polynomial. Thus $f \xrightarrow[*]{G} 0$ for all $f \in Ideal(G)$ which is equal to $NF_G(f) = \{0\}$.

From Buchberger's characterization of a Gröbner basis, one can deduce how to construct a Gröbner basis for a given ideal. The Gröbner basis for a given ideal is represented by a set of polynomials as generators. In the following, the basic version of Buchberger's algorithm is presented.

Algorithm 3.2 GRÖBNER-BASIS(F)

Input: A finite set of polynomials $F \subseteq K[\mathbf{x}]$.
Output: A Gröbner basis G for the ideal $Ideal(F)$.
$G := F$;
$B := \{S(f,g) \mid f,g \in F, f \neq g\}$; { $S(f,g)$ as defined in Equation (3.6)}
while $B \neq \emptyset$ **do**
  $B := B \setminus S_i$; {Remove one S-polynomial from B}
  $h :=$ NORMAL-FORM(S_i, G);
  if $h \neq 0$ **then**
    $B := B \cup \{S(f,h) \mid f \in G\}$;
    $G := G \cup \{h\}$;
  end if
end while
return G ;

EXAMPLE 3.8
This example computes the Gröbner basis for the following set F of polynomials using Buchberger's algorithm, Algorithm 3.2, while assuming lexicographic ordering:

$$F = \{xy^4 - 1, x^3 - y, -x^2 + y^5\}.$$

First, G is set equal to F. Then, the set B of S-polynomials throughout the algorithm is given by:

1) $B = \{-x^2 + y^5, xy^5 - y, -x + y^9\}$ and $G = \{xy^4 - 1, x^3 - y, -x^2 + y^5\}$

- $-x^2 + y^5 \xrightarrow[*]{G} 0$ by $-x^2 + y^5$
- $xy^5 - y \xrightarrow[*]{G} 0$ by $xy^4 - 1$
- $-x + y^9$ cannot be reduced by G

2) $B = \{y^{13} - 1, x^2y^9 - y, xy^9 - y^5\}$ and $G = \{xy^4 - 1, x^3 - y, -x^2 + y^5, -x + y^9\}$

- $x^2y^9 - y \xrightarrow[*]{G} 0$ by $xy^4 - 1$
- $xy^9 - y^5 \xrightarrow[*]{G} 0$ by $xy^4 - 1$
- $y^{13} - 1$ cannot be reduced by G

3) $B = \{x - y^9, y^{14} - y, -x^2 + y^{18}, -y^{22} + y^9\}$ and
$G = \{xy^4 - 1, x^3 - y, -x^2 + y^5, -x + y^9, y^{13} - 1\}$

- $x - y^9 \xrightarrow[*]{G} 0$ by $-x + y^9$
- $y^{14} - y \xrightarrow[*]{G} 0$ by $y^{13} - 1$
- $-x^2 + y^{18} \xrightarrow[*]{G} 0$ by $-x^2 + y^5$ and $y^{13} - 1$
- $-y^{22} + y^9 \xrightarrow[*]{G} 0$ by $y^{13} - 1$

4) $G = \{xy^4 - 1, x^3 - y, -x^2 + y^5, -x + y^9, y^{13} - 1\}$ is returned

Observe that the computation time of this algorithm highly depends on the sequence of the polynomials in F. Since G is not a Gröbner basis until the set B is empty, the output of the normal form algorithm is dependent on the ordering of the polynomials in G and thus is not unique. Therefore, the Gröbner basis as computed by this basic version of Buchberger's algorithm is not unique. Some additional conditions are needed to obtain a unique Gröbner basis.

DEFINITION 3.11
*A non-empty set $F \subseteq R[\mathbf{x}]$ is said to be **self-reduced** if each $f \in F$ is not reducible by $F \setminus \{f\}$ relative to some admissible ordering. The zero polynomial 0 is defined to be reducible by F for all F, and $f \neq 0$ is not reducible by the empty set.*

This definition implies that no self-reduced set contains the zero polynomial. Moreover, F is self-reduced if F is a singleton set and $F \neq \{0\}$. Also, if $a \in R$ and $a \in F$ then F is self-reduced iff $F = \{a\}$.

DEFINITION 3.12
*A finite non-empty set $G \subseteq F$ is a **reduced Gröbner basis** if all of the following conditions hold:*

(1) G is a Gröbner basis

(2) G is self-reduced

(3) $f \in G$ is monic, i.e. $hcoef(f) = 1$.

LEMMA 3.4 *Every ideal has a reduced Gröbner basis relative to some admissible ordering.*

A reduced Gröbner basis can be constructed from a Gröbner basis G by using Definition 3.11. For each $g \in G$, the normal form algorithm is applied with the set $G \setminus \{g\}$. If the result g' equals the zero polynomial, then g is discarded from G. Otherwise, g is replaced by g'. If no g can be further reduced by the set $G \setminus \{g\}$, each polynomial g is replaced by $g/hcoef(g)$ to make it monic.

THEOREM 3.5 *Reduced Gröbner bases are unique with respect to an admissible ordering.*

To discuss the decision problem for a system of polynomial equations, the relation between ideals and their corresponding zero sets have to be analyzed first.

Accoding to Corollary 3.1, the zero set of a set of polynomials $f_1, \cdots, f_p$ is equivalent to the zero set of the ideal J generated by the same set of polynomials: $ZERO(f_1, \cdots, f_p) = ZERO(J)$.

THEOREM 3.6 (Weak Nullstellensatz) *Let K be an algebraically closed field and let $J \subset K[\mathbf{x}] = K[x_1, \cdots, x_n]$ be an ideal satisfying $ZERO_K(J) = \emptyset$. Then, $J = K[\mathbf{x}]$.*

The Weak Nullstellensatz states that the only ideal which represents the empty zero set is the entire polynomial ring itself. But this theorem is only valid for algebraically closed fields. For example, consider the polynomials $f_1 = x^2 + 1$ and $f_2 = 1$ in $\mathbb{R}[\mathbf{x}]$ as well as the ideals generated by these polynomials $J_1 = Ideal(f_1)$ and $J_2 = Ideal(f_2)$. (Note that $\mathbb{R}$ is not algebraically closed.) Then, even though the two ideals are different, they both represent the empty zero set: $ZERO_{\mathbb{R}}(J_1) = ZERO_{\mathbb{R}}(J_2) = \emptyset$ because both polynomials do not have a solution in $\mathbb{R}$. However, observe that $ZERO_{\mathbb{C}}(J_1) \neq \emptyset$ but $ZERO_{\mathbb{C}}(J_2) = \emptyset$ since f_1 has a solution in $\mathbb{C}$. Thus, to prove if an ideal J is equal to $K[\mathbf{x}]$, one simply has to check whether the constant polynomial 1 is in J. If $1 \in J$, then, using Definition 2.53, $f = f \cdot 1 \in J$ for every $f \in K[\mathbf{x}]$ and thus $J = K[\mathbf{x}]$. For $K = \mathbb{C}$, the Weak Nullstellensatz implies that every system of polynomials that generate an ideal smaller than $\mathbb{C}[\mathbf{x}]$ has a common zero in $\mathbb{C}^n$.

Generally, a one-to-one mapping between ideals and their corresponding zero sets cannot be defined. Consider for example the two polynomials $f_1 = x$ and $f_2 = x^2$. Then, the ideals generated by these polynomials represent the same zero set: $ZERO(Ideal(f_1)) = ZERO(Ideal(f_2)) = \{0\}$. Hilbert's Nullstellensatz explains the reason for this.

THEOREM 3.7 (Hilbert's Nullstellensatz) *Let K be an algebraically closed field. Suppose $f, f_1, \cdots, f_m \in K[\mathbf{x}] = K[x_1, \cdots, x_n]$ are such that $f \in Ideal(ZERO(f_1, \cdots, f_m))$. Then, there exists an integer $s \geq 1$ such that $f^s \in Ideal(f_1, \cdots, f_m)$ and vice versa.*

The proof of Hilbert's Nullstellensatz is explained in detail because the trick used here is used again later in this section.

Proof:

Assume the polynomial f vanishes at every common zero of the set of polynomials $f_1, \cdots, f_m$. It has to be shown that there exists an integer $s \geq 1$ and polynomials $a_1, \cdots, a_m$ such that $f^s = \sum_{i=1}^{m} a_i \cdot f_i$. Therefore, a trick is used. The following ideal is considered:

$$\tilde{J} = Ideal(f_1, \cdots, f_m, y \cdot f - 1) \subset K[x_1, \cdots, x_n, y].$$

Obviously, the zero set of the ideal $\tilde{J}$, $ZERO(\tilde{J})$, is empty. Suppose a point $\alpha = (\alpha_1, \cdots, \alpha_n, \alpha_{n+1}) \in K^{n+1}$. Hence, if $(\alpha_1, \cdots, \alpha_n)$ is a common zero of $f_1, \cdots, f_m$, then also $f(\alpha_1, \cdots, \alpha_n) = 0$ which implies $y \cdot f - 1 = \alpha_{n+1} f(\alpha_1, \cdots, \alpha_n) - 1 = -1 \neq 0$ and thus $\alpha \notin ZERO(\tilde{J})$. In the other case, if $(\alpha_1, \cdots, \alpha_n)$ is not a common zero of $f_1, \cdots, f_m$,

then there exists at least one i, $1 \leq i \leq m$, for which $f_i(\alpha_1, \cdots, \alpha_n) \neq 0$ and therefore $\alpha \notin ZERO(\tilde{J})$ either. Therefore, $ZERO(\tilde{J}) = \emptyset$ can be concluded. Applying the Weak Nullstellensatz from Theorem 3.6 one can deduce from $ZERO(\tilde{J}) = \emptyset$ that $1 \in \tilde{J}$. It follows for all y that

$$1 = (\sum_{i=1}^{m} g_i(x_1, \cdots, x_n, y) f_i) + h(x_1, \cdots, x_n, y)(yf - 1),$$

where $g_i, h \in K[x_1, \cdots, x_n, y]$. The substitution $1/f(x_1, \cdots, x_n)$ for y makes the last term disappear:

$$1 = \sum_{i=1}^{m} g_i(x_1, \cdots, x_n, 1/f) f_i.$$

Multiplying both sides of the equation by f^s yields the desired, above stated, equation:

$$f^s = \sum_{i=1}^{m} a_i \cdot f_i.$$

□

Hilbert's Nullstellensatz means, if a polynomial f vanishes at all points of the zero set of a given Ideal J, then either it must be a member of the ideal J itself or some power of f must belong to J. The following lemma presents the condition for ideals that consist of all polynomials that vanish on the same zero set.

LEMMA 3.8 *Let Z be a zero set and $Ideal(Z)$ the ideal generated by this zero set. Then, if $f^s \in Ideal(Z)$, also $f \in Ideal(Z)$.*

Lemma 3.8 states that the ideal consisting of all polynomials that vanish on the same zero set is a radical ideal, as defined in Definition 2.54. One can construct a radical ideal from any ideal J by applying the ideal operation $Radical(J) = \sqrt{J}$ as given in Definition 2.59. Note that $J \subseteq \sqrt{J}$ since $f^1 \in J$ implies $f \in \sqrt{J}$ per definition. This leads to the condition that an ideal J is radical iff $J = \sqrt{J}$. Now, the Strong Nullstellensatz can be presented.

THEOREM 3.9 (Strong Nullstellensatz) *Let K be an algebraically closed field. If J is an ideal in $K[\mathbf{x}]$, then $Ideal(ZERO_K(J)) = \sqrt{J}$.*

The Strong Nullstellensatz finally provides the considered relation between ideals and their corresponding zero sets. There exists a one-to-one mapping between radical ideals and their corresponding zero sets, but not between ideals in general and their corresponding zero sets.

Now, the decision problem for a system of polynomial equations can be considered. Suppose the system of polynomial equations $f_1 = f_2 = \cdots = f_p = 0$ has to be analyzed regarding its common solution set. This is equivalent to the question to whether the zero set $ZERO(f_1, \cdots, f_p) = \emptyset$. As stated by Equation (3.2), the zero set of a set of polynomials is equivalent to the zero set of the ideal generated by the same set of polynomials, i.e. $ZERO(f_1, \cdots, f_p) = ZERO(Ideal(f_1, \cdots, f_p))$. Applying the Weak Nullstellensatz for $\mathbb{C}$ from Theorem 3.6, the question whether the zero set $ZERO_{\mathbb{C}}(f_1, \cdots, f_p) = \emptyset$ amounts to the question: "Is $1 \in Ideal(f_1, \cdots, f_p)$?" Since any polynomial equation consisting of a

constant polynomial does not have any solutions, the zero set of the ideal containing such a polynomial is empty and therefore the corresponding system of polynomial equations is inconsistent with respect to $\mathbb{C}$. Thus in order to decide the decision problem for $\mathbb{C}$ for a system of polynomial equations, the **Ideal Membership question** "Is element f in $Ideal(f_1, \cdots, f_p)$?" has to be answered for $f = 1$.

LEMMA 3.10 (Ideal Membership) *Let K be an arbitrary field and let $J \subset K[\mathbf{x}]$ be the ideal generated by $f_1, \cdots, f_m$. Assume G to be the Gröbner basis of J. Then $f \in J$ iff $f \xrightarrow[*]{G} 0$, i.e. $NF_G(f) = 0$.*

According to Lemma 3.10, one can determine whether an element f is a member of the ideal $Ideal(f_1, \cdots, f_p)$, by first computing the Gröbner basis G of $Ideal(f_1, \cdots, f_p)$ and then checking the normal form of f with respect to the Gröbner basis G for equality with zero. For $f = 1$, one can simply test if the Gröbner basis G contains a nonzero constant instead of applying the normal form algorithm. In terms of a reduced Gröbner basis, it is even more convenient since $\{1\}$ is the only reduced Gröbner basis for the ideal $Ideal(1) = \mathbb{C}[\mathbf{x}]$. In the following, the consistency algorithm for the decision problem for any algebraically closed field is summarized:

Algorithm 3.3 CONSISTENT(F)

Input: A finite set of polynomials $F \subseteq K[\mathbf{x}]$.
Output: A boolean variable indicating the consistency state of the set of polynomial equations.
$c := TRUE$;
$i = 1$;
$G :=$ GRÖBNER-BASIS(F); {with respect to any ordering}
G-tuple := N-TUPLE(G); {$g_i \in$ *G-tuple*}
while (c) **and** ($i \leq |G|$) **do**
  if ($g_i = nonzero\ constant$) **then**
    $c := FALSE$**;**
  end if
  i := i +1;
end while
return c **;**

Note that if the field K is not algebraically closed, the decision problem can only be answered in one direction: If $\{1\}$ is a reduced Gröbner basis for the ideal $Ideal(f_1, \cdots, f_p) \subseteq K[\mathbf{x}]$, then the system of polynomial equations $f_1 = \cdots = f_p = 0$ has no common solution. The reverse is not true. Consider the fields $\mathbb{R}$ and $\mathbb{C}$. If a system of polynomial equations does not have any common complex zeros, it does not have any common real zeros either. But on the other hand, if a system of polynomial equations does not have any common real zeros, it can still have common complex zeros since $\mathbb{R}$ is not algebraically closed.

Now, the relationship between Gaussian elimination and Buchberger's algorithm is explained as well as the connection between the Euclidean and Buchberger's algorithm is indicated.

In the same manner, as the S-polynomials provide the cancellation of the head monomials to get a simpler set of equations in the non-linear multivariate case, the Gaussian

elimination performs the same task in the linear multivariate case. The reduced polynomials received from the row echelon form of the matrix of coefficients after Gaussian elimination are the same as the Gröbner basis generators computed from Buchberger's algorithm using lexicographical ordering.

In the Euclidean algorithm, the greatest common divisor of two non-linear univariate polynomials is computed as the generator for the ideal of these two polynomials. Therefore, the DIVISON algorithm (Algorithm 2.2), is applied, which corresponds to the NORMALFORM algorithm (Algorithm 3.1), in Buchberger's algorithm for the non-linar multivariate case. The ideal membership problem for the univariate case: The question "Is $f \in K[x]$ an element of $Ideal(f_1, \cdots, f_p)$?" can be determined using the Euclidean algorithm. First, the greatest common divisior is computed to find the single generator g of the ideal. Then, $f \in Ideal(f_1, \cdots, f_p)$ equals to $f \in Ideal(g)$, which can be answered by the DIVISON algorithm. $f = qg + r$ is element of $Ideal(g)$ if and only if $r = 0$.

For the application of geometical theorem proving, one takes advantage of the relationship between geometry and algebra. As explained before in this section, there exists a one-to-one mapping between radical ideals and their corresponding zero sets. To prove a geometrical theorem means to verify questions of the form:

$$\forall \mathbf{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n : \text{if } h_1(\mathbf{x}) = 0, \cdots, h_r(\mathbf{x}) = 0 \text{ then } c(\mathbf{x}) = 0, \tag{3.7}$$

where $h_1, \cdots, h_r$ denote the hypotheses and c represents the conclusion. All hypotheses and the conclusion have to be expressed as polynomial expressions. The answer to this question can be found by replacing $\mathbb{R}$ by $\mathbb{C}$ and by solving the radical membership question for c, i.e $c \in Radical(h_1, \cdots, h_r)$.

LEMMA 3.11 (Radical Membership) *Let K be an arbitrary field. Let $J \subset K[x_1, \cdots, x_n]$ be the ideal generated by $f_1, \cdots, f_p$. Then, $f \in \sqrt{J}$ iff the constant polynomial 1 belongs to the ideal $\tilde{J} = Ideal(f_1, \cdots, f_p, y \cdot f - 1) \subset K[x_1, \cdots, x_n, y]$.*

Proof:
Using the proof of Hilbert's Nullstellensatz, one can deduce from $J \subset \tilde{J}$ that $f^s \in J$ for some s, which in turn implies $f \in \sqrt{J}$. In the other direction, if $f \in \sqrt{J}$ then $f^s \in J \subset \tilde{J}$ for some s.

□

In order to prove a geometrical theorem it is not possible to apply simply the consistency question on the hypotheses and the conclusion by using the ideal membership problem on the ideal $I_1 = Ideal(h_1, \cdots, h_r, c)$. This would answer the question, whether there exists a common solution for the system of polynomial equations $h_1, \cdots, h_r, c$, but the question is, whether c vanishes on all common solutions of $h_1, \cdots, h_r$. Equally, a geometrical theorem cannot be proved by applying the ideal membership problem of the polynomial c on the ideal $I_2 = Ideal(h_1, \cdots, h_r)$. The consistency question can fail even though c might vanish on the same zero set because ideals and zero sets do not have a one-to-one correspondence, and in this case $c^s \in I_2$ but not $c \in I_2$ as stated in the Strong Nullstellensatz in Theorem 3.9 and Hilbert's Nullstellensatz in Theorem 3.7.

Therefore, a geometrical theorem can be proved using the radical membership problem. One has to compute the Gröbner basis G for the ideal $J = Ideal(h_1, \cdots, h_r, y \cdot c - 1)$ and

has to decide whether 1 belongs to the ideal J by checking if G contains any nonconstant polynomial. Alternatively, the reduced Gröbner basis could be used, which then has to be tested for equality with $\{1\}$. The ideal membership question could also be used to test whether $f \in \sqrt{J}$ by checking if $f^s \in J$ for any integer $s \geq 1$. However, this turns out to be quite inefficient.

As already remarked for the consistency problem, this method of geometrical theorem proving can only be used in one direction since the problem in $\mathbb{R}$ has to be shifted to the algebraically closed field $\mathbb{C}$. This means, if a geometrical theorem holds over $\mathbb{C}$ then it also holds over $\mathbb{R}$, but the reverse it not true in general. Therefore, only confirmations but no refutations of geometrical theorems are possible. However, most geometrical theorems are generally true over $\mathbb{C}$, [Buc89].

It is also necessary to treat degenerate situations since most geometrical theorems are indeed true for the general case but they may be false for degenerate situations. Degenerate situations are situations when, for example, a radius or an angle becomes zero. If a degenerate situation can be expressed as a polynomial inequation d, $d(x_1, \cdots, x_n) \neq 0$, a new variable, also called slack variable, can be introduced to transform it into a polynomial equation, which is considered as an additional condition. The following lemma describes this transformation.

LEMMA 3.12 *The satisfiability of $f \neq 0$ is equivalent to the satisfiability of $uf - 1 = 0$, where u is a new variable not appearing in f.*

Therefore, a geometrical theorem including degenerate situations, like:

$$\forall \mathbf{x}((h_1 = 0 \wedge \cdots \wedge h_r = 0 \wedge d_1 \neq 0 \wedge \cdots \wedge d_t \neq 0) \Longrightarrow c = 0), \tag{3.8}$$

is equivalent to

$$\exists \mathbf{x}, \mathbf{u}, z((h_1 = 0 \wedge \cdots \wedge h_r = 0 \wedge u_1 \cdot d_1 = 1 \wedge \cdots \wedge u_t \cdot d_t = 1 \wedge v \cdot c = 1), \tag{3.9}$$

which can be solved by testing whether $\{1\}$ is the reduced Gröbner basis of the ideal $Ideal(h_1, \cdots, h_r, u_1 \cdot d_1 - 1, \cdots, u_t \cdot d_t - 1, v \cdot c - 1)$. Again, any polynomial inequation can be transformed to a polynomial equation using slack variables and thus can be handled by the Gröbner basis radical membership method. The equation $u \cdot d = 1$ assures that the polynomial expression d never becomes zero. Inequalities involving ordering relations such as $<, >, \leq$ and $\geq$ cannot be treated with this method. For a more detailed investigation in geometric reasoning, one can refer to [KM89]. In context with Gröbner basis, several discussions can be found in [Kap86] and [KS86].

Coming back to the consistency algorithm, Algorithm 3.3, for a system of polynomial equations, inequations can also be included herein by applying the transformation on inequations to equations as explained in Lemma 3.12.

In order to extend the validity of the consistency problem from $\mathbb{C}$ to $\mathbb{R}$, one could try to compute the set of solutions of the system of polynomial equations explicitly and see if it contains any real solution. Subsequently, three approaches are presented for a system of polynomial equations with finitely many solutions using Gröbner basis computation: the

elimination method, the univariate-polynomial method and the side condition method. All methods can only be applied in case the considered ideal is zero-dimensional. This will be explained in the following.

A set $S \subseteq R[\mathbf{x}]$ is **zero-dimensional** iff its zero set $ZERO(S) \subseteq R^n$ is finite. As an example for a set of polynomials which is zero-dimensional consider $S = \{x_1^2 - 1, x_2^2, \cdots, x_n^2\}$. Then, obviously $ZERO(S)$ has the following two n-tuples as roots: $(1, 0, \cdots, 0)$ and $(-1, 0, \cdots, 0)$.

THEOREM 3.13 *Let $J \subseteq K[x_1, \cdots, x_n]$ be an ideal. Then J is zero-dimensional iff for each $i = 1, \cdots, n$, $J \cap K[x_i] \neq \emptyset$. In general, the dimension of J is the largest cardinality of a set $\mathbf{y} \subseteq \mathbf{x}$ with $\mathbf{x} = (x_1, \cdots, x_n)$ such that $J \cap K[\mathbf{y}] = Ideal(0)$.*

The finite solvability of a set of polynomial equations can be tested easily using Gröbner basis computation as explained in the following theorem and lemma.

THEOREM 3.14 *Let F be a set of polynomials in $K[x_1, \cdots, x_n]$ and let G be a Gröbner basis for $Ideal(F)$. Then, G and consequently F are zero-dimensional iff for each $i = 1, \cdots, n$ there exists a $g_i \in G$ with $hterm(g_i) \in PP(x_i)$, where $PP(x_i)$ is the set of power products over the variable x_i as defined in Section 2.2 by Definition 2.20. This means that G and F have finitely many solutions iff for all i, $1 \leq i \leq n$, there exists a $k > 0$ such that ax_i^k, $a \in K$, is the head monomial of a polynomial in G.*

In case G is a reduced Gröbner basis, the head monomials are monic and thus a k has to be found such that x_i^k is the head monomial of a polynomial in G. Theorem 3.14 leads to a convenient property of the Gröbner basis.

LEMMA 3.15 *Let F be a set of polynomials in $K[x_1, \cdots, x_n]$. Assume the following precendence on the variables: $x_1 < \cdots < x_n$ and let $G = \{g_1, \cdots, g_q\}$ be a Gröbner basis for $Ideal(F)$ with respect to lexicographic ordering, $>_{lex}$. If F has finitely many solutions, then G has **triangular form** defined as:*

$$\begin{aligned} g_1, \cdots, g_{s_1} &\in K[x_1] \\ g_{s_1}, \cdots, g_{s_2} &\in K[x_1, x_2] \\ &\vdots \\ g_{s_i}, \cdots, g_{s_{i+1}} &\in K[x_1, \cdots, x_{n-1}] \\ g_{s_{i+1}}, \cdots, g_q &\in K[x_1, \cdots, x_n]. \end{aligned} \tag{3.10}$$

The next theorem combines the finite solvability with the dimension of an ideal and the triangular form of the Gröbner basis.

THEOREM 3.16 *Let $F \subset K[\mathbf{x}]$ be a system of polynomials. Then the following three statements are equivalent:*

1) F is finitely solvable.

2) $Ideal(F)$ is a proper zero-dimensional ideal.

3) If G is a Gröbner basis of $Ideal(F)$ with respect to $>_{lex}$, then G can be expressed in triangular from.

Now, the **elimination method** for computing the real zeros of a system of polynomials F is explained. According to Lemma 3.15, the Gröbner basis G has triangular form iff F has finitely many zeros. This means that at least one g_i of the Gröbner basis is a univariate polynomial in the last variable with respect to the precendence on the variables. The roots of the univariate polynomial are determined and propagated as a super set of solutions of x_1 to the polynomials in $K[x_1, x_2]$. This again leads to univariate polynomials, in this case in the variable x_2. The zeros of theses polynomials build a super set of two-dimensional partial solutions, which are propagated to the next higher dimensional polynomials to increment the solutions by one dimension. The polynomials are handled sequentially such that the set of partial solutions is extended or confirmed. Non-valid solutions are omitted when the validation of a partial solution or the elimination of another variable fails. This procedure is repeated until all g_i of the Gröbner basis has been solved and thus the zero set for F is completed.

EXAMPLE 3.9
Given the following system of polynomials $F = \{f_1, f_2\}$:

$$\begin{aligned} f_1 &= x^2 + xy + 2x + y - 1 \\ f_2 &= xy - x + y^2 - y. \end{aligned} \tag{3.11}$$

The Gröbner basis G is calculated using lexicographic ordering and the precendence $y > x$ on the variables:

$$\begin{aligned} g_1 &= x^2 - 1 \\ g_2 &= yx - y - x + 1 \\ g_3 &= y^2 + 3y + 2x - 2. \end{aligned} \tag{3.12}$$

EXAMPLE 3.10 (Elimination method)
Now, the elimination method is used to solve the system of polynomials in (3.11) with the Gröbner basis G in (3.12). First, the univariate polynomial g_1 is solved which yields $x = 1$ or $x = -1$: $ZERO(g_1) = \{1, -1\}$. Both solutions are propagated to g_2. $x = 1$ provides the solution $(1, y)$ while $x = -1$ results in $(-1, 1)$: $ZERO(\{g_1, g_2\}) = \{(1, y), (-1, 1)\}$. This set of solutions is inserted into g_3. The first solution is refined and gives $\{(1, -3), (1, 0)\}$ and the second one is confirmed valid: $ZERO(F) = ZERO(\{g_1, g_2, g_3\}) = \{(1, -3), (1, 0), (1, -1)\}$.

An implementation of the elimination method is given in [Mon96]. The procedure which solves the univariate polynomials uses a numerical method to determine the zeros of the polynomials, which might yield a possible loss of solutions. This can be improved using so-called real algebraic numbers. Different representations of real algebraic numbers and computation methods with algebraic numbers are explained in detail in Section 4.2.

The second method for extracting the roots of a system of polynomials F is called **univariate-polynomial method**. This method first computes the Gröbner Basis G for the ideal $Ideal(F)$ and then constructs univariate polynomials as the smallest polynomials $p_i \in Ideal(F)$ in x_i using an algorithm from Buchberger [Buc85]. Each univariate

polynomial p_i can be solved independantly. The zero set of F is obtained by building the super set of all possible combinations of n-tuples composed of a solution of each p_i, reduced by those n-tuples which cannot be verified by all g_i of G.

The basis of Buchberger's algorithm for constructing the univariate polynomials based on a given Gröbner Basis G, in case F is finitely solvable, is given by Theorem 3.17.

THEOREM 3.17 *Let G be a Gröbner basis with respect to lexicographic ordering $\underset{lex}{\leq}$ where $x_1 \underset{lex}{\leq} \cdots \underset{lex}{\leq} x_n$. Then, G is zero-dimensional iff for each $i = 1, \cdots, n$ there exists a polynomial $p_i \in G$ such that $p_i \in K[x_i]$.*

The next algorithm presents the algorithm from Buchberger to construct univariate polynomials for a given Gröbner basis.

Algorithm 3.4 UNIVARIATE-POLYNOMIAL(G, x)

Input: A Gröbner basis $G \subseteq K[x_1, \cdots, x_n]$ and a variable x_i.
Output: The smallest univariate polynomial in the variable x_i whose solutions contain at least those values that occur in the n-tuples of $ZERO(G)$ for x_i.
$expo := 0$;
$p_{expo} = 1$;
while $\neg((a_0, \cdots, a_{expo}) \neq (0, \cdots, 0)$ such that $\sum_{i=0}^{expo} a_i p_i = 0)$ **do**
  $expo := expo + 1$;
  $p_{expo} :=$ NORMAL-FORM$(x \cdot p_{expo-1}, G)$;
end while
return $\frac{1}{a_{expo}} \sum_{i=0}^{expo} a_i {x_i}^{expo}$;

EXAMPLE 3.11 (Univariate-polynomial method)
Now, the univariate-polynomial method is used to solve the system of polynomials F in (3.11) with the Gröbner basis G in (3.12). First, a univariate polynomial has to be computed for each variable. Therefore, the univariate-polynomial Algorithm 3.4 is applied for x and for y. The calculation of UNIVARIATE-POLYNOMIAL(G, y) is presented stepwise in Table 3.1.

$expo$	$x \cdot p_{expo-1}$	p_{expo}	$\sum_{i=0}^{expo} a_i p_i$
0	-	1	a_0
1	y	y	$a_0 + a_1 y$
2	y^2	$2 - 2x - 3y$	$a_0 + a_1 y + a_2(2 - 2x - 3y)$
3	$2y - 2yx - 3y^2$	$-4 + 4x + 9y$	$a_0 + a_1 y + a_2(2 - 2x - 3y) + a_3(-4 + 4x + 9y)$

Table 3.1: Computation of UNIVARIATE-POLYNOMIAL (G, y).

The algorithm terminates because a non-zero solution can be found for $\sum_{i=0}^{expo} a_i p_i$, for example: $a_0 = 0$, $a_1 = -3$, $a_2 = 2$ and $a_3 = 1$. The univariate polynomial for y is

returned as $\frac{1}{a_{expo}} \sum_{i=0}^{expo} a_i {y_i}^{expo} = y^3 + 2y^2 - 3y =: u_1$. The computation of UNIVARIATE-POLYNOMIAL(G, x) provides the univariate polynomial: $u_2 := x^2 - 1$. The zeros in y of u_1 are $\{0, 1, -3\}$, the roots in x of u_2 are given by $\{1, -1\}$. This leads to the super set of possible solutions of F, consisting of the cross product of all zeros in x and y: $\{(1,0), (1,1), (1,-3), (-1,0), (-1,1), (-1,-3)\}$. Each n-tuple has to be verified by every polynomial g_i in G as computed in (3.12). Non-valid solutions are removed. For example, inserting $(-1, 0)$ into g_3 results in -4 and not in 0. Finally, computing the complete zero set of the system of polynomials yields in $ZERO(F) = \{(1,0), (-1,1), (1,-3)\}$.

In general, the advantage of the univariate-polynomial method is to remain symbolic until the computation of the zeros of each polynomial. Introducing real algebraic numbers to this method, one could receive further improvements. As mentioned above, different representations of real algebraic numbers and computation methods with algebraic numbers are presented in detail in Section 4.2. However, the disadvantage of the univariate-polynomial method is to have too large super sets due to a combinatorial explosion. This can be reduced to some extend using Algorithm 3.5 to test the valitity of the super set of solutions [Mon96]. Algorithm 3.5 takes advantage of the triangular form of a Gröbner basis when computed with respect to lexicographic ordering.

Algorithm 3.5 starts with an ascendingly sorting procedure in order to work on a Gröbner basis in triangular form with respect to lexicographic ordering. Then, the first univariate polynomials of G are obtained in the smallest variable, the first variable in V, and are omitted for the test of validity since the solutions are their own zeros. The solutions are created incrementally as the carthesian product of the previous carthesian product and the next variable in order. The carthesian product represents the set of valid partial solutions at the end of each outer FOR loop which goes through every g_i of G starting from equations with two variables. The Procedure VARIABLES returns the set of variables for a chosen generator of G. Thus, this algorithm diminishes the combinatorial exposion of the carthesian product because it applies the latter only to the valid set of the previous carthesian product.

Howevever, the eliminaton method is more efficient because the incrementality avoids the propagation of non-valid partial solutions and because the usage of partial solutions restricts combinatorial explosions.

The third method in order to count the real zeros of a system of polynomials F, the **side condition method**, is discussed now. The side condition method is a simple case of the method introduced by Pedersen, Roy and Szpirglas [PRS93] which counts the real zeros of a system of polynomials F under one side condition based on Gröbner bases computations. Again, this method is also constrained to zero-dimensional ideals, which means that the zero set of the polynomial system has to be finite. Given a set of polynomials $F = \{f_1, \cdots, f_p\}$ and a side condition h in $K[x_1, \cdots, x_n]$ the method of Pedersen, Roy and Szpirglas counts the zeros in $\mathbb{R}^n$ such that $f_i(x_1, \cdots, x_n) = 0$ holds for all $1 \leq i \leq p$ and $h(x_1, \cdots, x_n)\, \rho\, 0$, where $\rho \in \{<, =, >\}$. The side condition method counts the real zeros of a system of polynomials F unter the trivial side condition $h = 1$.

Algorithm 3.5 VALIDATE-ZEROSET(G, V, S)

Input: A Gröbner basis $G \subseteq K[x_1, \cdots, x_n]$ computed with lexicographic ordering, a tuple V of variables presenting the ascending precendence of them, and a tuple S containing the sets of zeros for each variable in V.
Output: The zero set of G consisting of all valid n-tuples.
$i := 1$; $s := 2$;
G-tuple := SORT-ASC(N-TUPLE$(G), V)$; {*G-tuple* is sorted ascendingly w.r.t. the variable precendence}
while VARIABLES$(g_i) = \{V_1\}$ **do**
  $i := i + 1$;
end while
$CarthesianProduct := S_1 \times S_2$;
$CP\text{-}help := CarthesianProduct$;
$SetVar :=$ VARIABLES(g_i);
for $j = i$ to $|G|$ **do**
  if VARIABLES$(g_j) \neq SetVar$ **then**
    $s := s + 1$;
    $CarthesianProduct := CarthesianProduct \times S_s$;
    $CP\text{-}help := CarthesianProduct$;
  end if
  for $k = 1$ to $|CarthesianProduct|$ **do**
    if $g_j(CarthesianProduct_k) \neq 0$ **then**
      $CP\text{-}help := CP\text{-}help \setminus CarthesianProduct_k$;
    end if
  end for
  $CarthesianProduct := CP\text{-}help$;
end for
return $CarthesianProduct$;

First, the method of Pedersen, Roy and Szpirglas is explained. In general, this method is able to answer the decision problem of a system of non-linear multivariate equations and one inequation for the reals. Then, an example is introduced to illustrate the application of the side condition method to count the real zeros of a system of polynomials F under the side condition $h = 1$.

The following Theorem provides the basics of the method of Pedersen, Roy and Szpirglas. It considers the quotient ring $K[\mathbf{x}]/J$ of the polynomial ring $K[\mathbf{x}]$, as defined by Definition 2.60, as a vector space over the field K of coefficients.

THEOREM 3.18 *Let K be an ordered field. Let R be a real closed field such that $K \subset R$ and let C be its algebraic closure. Take J to be the ideal in $K[\mathbf{x}]$ generated by the set of polynomials $F = \{f_1, \cdots, f_p\}$ and fix $B := \{v_1, \cdots, v_b\}$ to be the basis of the K-vector space $K[\mathbf{x}]/J$. For any polynomial $h \in K[\mathbf{x}]$ define the matrix Q_h as follows:*

$$Q_h := (trace(m_{h \cdot v_i \cdot v_j}))_{1 \leq i,j \leq b}, \tag{3.13}$$

where $m_f : K[\mathbf{x}]/J \to K[\mathbf{x}]/J$ is the vector space endomorphism induced by multiplication by f in the quotient ring $K[\mathbf{x}]/J$. Then it holds:

(1) The matrix Q_h is real, quadratic, and symmetric.

(2) $rank(Q_h) = |\{\mathbf{x} \in ZERO_C(J) \mid h(\mathbf{x}) \neq 0\}|$.

(3) $signature(Q_h) = |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) > 0\}| - |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) < 0\}|$.

Remember that the trace of a matrix is calculated as the sum of its diagonal elements. Rank and signature in this special case can be computed according to the subsequent definition.

DEFINITION 3.13
Let Q be a real, symmetric $b \times b$ matrix. Let π_Q denote the number of positive eigenvalues of Q counted with multiplicity and let ν_Q indicate the number of negative eigenvalues of Q also counted with multiciplity. Then, rank and signature of Q can be determined as follows:

(1) $rank(Q) = \pi_Q + \nu_Q$.

(2) $signature(Q) = \pi_Q - \nu_Q$.

Note that all eigenvalues of a real, symmetric matrix are always real. The eigenvalues of the matrix Q are given by the zeros of the associated characteristic polynomial λ_Q, which is defined as $\lambda_Q = det(Q - t \cdot I)$ and where I denotes the identity matrix. Therefore, the zeros of the characteristic polynomial are also real. The number of positive and negative eigenvalues can then be determined using Descartes' rule of sign, which has been presented in Theorem 2.10 in Section 2.3. Suppose the characteristic polynomial to be $\lambda_Q(t) = \sum_{i=0}^{b} a_i t_i$. Then, the number of positive eigenvalues of Q are equal to the number of sign variations in the sequence of coefficients: $\pi_Q = Var(a_b, a_{b-1}, \cdots, a_1, a_0)$. Similarily, the number of negative eigenvalues of Q can be evaluated from the sequence of coefficients of $\lambda_Q(-t)$: $\nu_Q = Var(a_b, -a_{b-1}, a_{b-2}, \cdots, (-1)^b a_0)$.

It can be easily deduced from Theorem 3.18, that the number of solutions in C of the set of polynomials $F = \{f_1, \cdots, f_p\}$ is the rank of the matrix Q_1: $rank(Q_1) = |ZERO_C(J)|$. Equally, one can show that the number of real zeros is determined by the signature of Q_1: $signature(Q_1) = |ZERO_R(J)|$. Another special case can be obtained from this theorem. It can be derived that $signature(Q_{h^2}) = |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) \neq 0\}|$. Combining these two special cases with the general case, one can set up the following equation, whose solution presents the number of real zeros under one given side condition h:

$$\begin{bmatrix} 1 & 1 & 1 \\ -1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) < 0\}| \\ |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) = 0\}| \\ |\{\mathbf{x} \in ZERO_R(J) \mid h(\mathbf{x}) > 0\}| \end{bmatrix} = \begin{bmatrix} signature(Q_1) \\ signature(Q_h) \\ signature(Q_{h^2}) \end{bmatrix}. \tag{3.14}$$

Equation (3.14) is uniquely solvable since the determinant of its first matrix is not zero. Note that this equation is similar to Equation (2.39) for the univariate case from Section 2.3, which is based on Sturm Sequences. An outline of the algorithm is presented in the following. It takes the set of polynomials F and the side condition h in $K[\mathbf{x}]$ as input and computes the number of common real zeros of F for $h(\mathbf{x}) < 0$, $h(\mathbf{x}) = 0$ and $h(\mathbf{x}) > 0$ according to Equation (3.14).
An implementation of this algorithm as well as a detailed explanation of the mathematical background (such as bilinear form, commutative K-algebras, structure constants) can

Algorithm 3.6 REAL-ZEROSET(F, h)

Input: A system of polynomial equations $F = \{f_1, \cdots, f_p\} \subseteq K[x_1, \cdots, x_n]$ and the polynomial of one side condition h.
Output: The number of common real zeros of F for $h(\mathbf{x}) < 0$, $h(\mathbf{x}) = 0$ and $h(\mathbf{x}) > 0$
G = GRÖBNER-BASIS(F); {with respect to any ordering}
$B = \{v_1, \cdots, v_b\}$: compute basis of the ideal quotient ring $K[\mathbf{x}]/J$;
$Q_h = (trace(m_{h \cdot v_i \cdot v_j}))_{1 \leq i,j \leq b}$: compute matrix Q_h for $h = 1$, h and h^2;
$s := (signature(Q_1), signature(Q_h), signature(Q_{h^2}))$;
$zerovector := (|\{ZERO_R(J) \mid h < 0\}|, |\{ZERO_R(J) \mid h = 0\}|, |\{ZERO_R(J) \mid h > 0\}|)$;
{using Corollary (3.14)}
return *zerovector*;

be found in [Lip93]. The applied method can be extended to count the real zeros of a system of polynomials under finitely many side conditions. Given a set of polynomials $F = \{f_1, \cdots, f_p\}$ and finitely many side conditions $h_1, \cdots, h_s$ in $K[x_1, \cdots, x_n]$, the extended method counts the zeros in $\mathbb{R}^n$ for which $f_i(x_1, \cdots, x_n) = 0$ for all $1 \leq i \leq p$ and $h_j(x_1, \cdots, x_n)\, \rho\, 0$ for all $1 \leq j \leq s$ and where $\rho \in \{<, =, >\}$. The extended method uses the work of Ben-Or, Kozen and Reif, [BOKR86] and has been invented by Weispfenning, [Wei93] or revised in [Wei98]. An implemenation, called QERRC, of this method is given in [Dol94]. The latter two references also address parametric problems using comprehensive Gröbner bases. An extensive study of comprehensive Gröbner bases can be found in [Wei92].

Again, in order to answer the decision problem it is sufficient to compute the signature of the matrix Q_1, which counts the number of real zeros of a system of polynomial equations when no side condition is given. The following example, which has been taken from [Pet97], explains the different steps to answer this question.

EXAMPLE 3.12
Consider the polynomial system $F = \{f_1, f_2\}$ in $\mathbb{Q}[x_1, x_2]$, where f_1 and f_2 are given by:

$$\begin{aligned} f_1 &= x_1x_2 - x_1^3 + x_2^2 - 2x_1^2x_2 \\ f_2 &= 2x_1^2 - x_2^2 + x_1x_2. \end{aligned} \tag{3.15}$$

Let J be the ideal generated by the polynomials f_1 and f_2. The first step is to compute a Gröbner basis with respect to an arbitrary monomial ordering. In this case, the reverse lexicographic ordering $\underset{rev}{\geq}$ as defined in Definition 2.36 in Section 2.2, is applied. Thus, for the Gröbner basis G it holds that:

$$G = \{5x_2^4 - 32x_2^3 + 32x_1x_2 + 32x_2^2, 2x_1^2 - x_2^2 + x_1x_2, 4x_1x_2 + 4x_2^2 + x_1x_2^2 - 3x_2^3\}.$$

Next, the basis $B = \{v_1, \cdots, v_b\}$ of the $\mathbb{Q}$-vector space $\mathbb{Q}[x_1, x_2]/J$ is calculated. Basically, the $\mathbb{Q}$-vector of any polynomial is determined by computing its normal form and then taking the coefficients of the monomials as vector elements. The normal form is obtained by using the NORMAL-FORM algorithm with respect to G and a fixed monomial ordering. In such a normal form only those monomials whose exponent vector appear below the exponent vector of the head monomials of the Gröbner basis generators can occur. Assume

$T = \{x_1^{e_1} \cdots x_n^{e_n} \,|\, (e_1, \cdots, e_n) \in \mathbb{N}^n\}$ to be the set of all terms in $\mathbb{Q}[\mathbf{x}]$. Then, the basis B of the $\mathbb{Q}$-vector space $\mathbb{Q}[x_1, \cdots, x_n]/J$ is given by $B = T \setminus hterm(J) = T \setminus hterm(Ideal(G))$. In this example, $hterm(G) = \{x_2^4, x_1^2, x_1x_2^2\}$ and $hterm(Ideal(G))$ consists of $hterm(G)$ and all higher terms with respect to $\underset{rev}{\geq}$. Therefore for the basis B it holds that $B = \{1, x_2, x_2^2, x_2^3, x_1, x_1x_2\}$. Accoding to Equation (3.14), the matrix Q_1 is constructed by $Q_1 := (trace(m_{v_i \cdot v_j}))_{1 \leq i,j \leq b}$. In order to compute the matrices $m_{v_i \cdot v_j}$ for all $1 \leq i, j \leq b$, one has to build a matrix for each variable x_p for $1 \leq p \leq n$ by computing the normal form of each term in B multiplied by x_p. In this case, one can verify that the normal forms for x_1 are:

$$x_1, x_1x_2, -4x_1x_2 - 4x_2^2 + 3x_2^3, \frac{16}{5}(x_2^3 - x_1x_2 - x_2^2), \frac{1}{2}(x_2^2 - x_1x_2), 2x_1x_2 + 2x_2^2 - x_2^3,$$

and those for x_2 are:

$$x_2, x_2^2, x_2^3, \frac{32}{5}(x_2^3 - x_1x_2 - x_2^2), x_1x_2, -4x_1x_2 - 4x_2^2 + 3x_2^3.$$

This lead to the following matrices:

$$m_{x_1} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -4 & -\frac{16}{5} & \frac{1}{2} & 2 \\ 0 & 0 & 3 & \frac{16}{5} & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & -\frac{16}{5} & -\frac{1}{2} & 2 \end{bmatrix}, \qquad m_{x_2} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -\frac{32}{5} & 0 & -4 \\ 0 & 0 & 1 & \frac{32}{5} & 0 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{32}{5} & 1 & -4 \end{bmatrix}.$$

The matrices $m_{v_i \cdot v_j}$ for all $1 \leq i, j \leq b$ can now be easily obtained from the matrices m_{x_1} and m_{x_2} since the vector space endomorphism $m_f : K[\mathbf{x}]/J \to K[\mathbf{x}]/J$ is such that $m_f m_g = m_{fg}$. (This extends to any polynomial h by summing the matrices computed for each monomial of the normal form of h.) For example, the matrix $m_{v_2v_6}$ is then equal to:

$$m_{v_2v_6} = m_{v_2}m_{v_6} = m_{x_1}m_{x_2}^2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -\frac{16}{5} & -\frac{192}{5} & -\frac{2304}{125} & -\frac{8}{5} & -\frac{96}{5} \\ 3 & \frac{16}{5} & \frac{192}{5} & \frac{2304}{125} & \frac{8}{5} & \frac{96}{5} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -4 & -\frac{16}{5} & -\frac{192}{5} & -\frac{2304}{125} & -\frac{8}{5} & -\frac{96}{5} \end{bmatrix}.$$

The trace of the matrix $m_{v_2v_6}$ can then be calculated to be $trace(m_{v_2v_6}) = \frac{864}{125}$, which is the entry of row 2 and column 6 in the matrix Q_1. Finally, Q_1 is equal to:

$$Q_1 = \frac{6}{15625} \begin{bmatrix} 15625 & 6250 & 15000 & 36000 & 3125 & 7500 \\ 6250 & 15000 & 36000 & 86400 & 7500 & 18000 \\ 15000 & 36000 & 86400 & 207360 & 18000 & 43200 \\ 36000 & 86400 & 207360 & 497664 & 43200 & 103680 \\ 3125 & 7500 & 18000 & 43200 & 3750 & 9000 \\ 7500 & 18000 & 43200 & 103680 & 9000 & 21600 \end{bmatrix}.$$

The rank of Q_1 is 2. As mentioned above, Equation (3.14) reduces for the decision problem to $signature(Q_1) = |ZERO_R(J)|$. This means, computing the signature of Q_1 provides the

number of real solutions of F, as defined in Equation (3.15). According to Definition 3.13, the signature can be determined by calculating π_{Q_1} and ν_{Q_1}, the number of positive and respectively negative eigenvalues of matrix Q_1: $signature(Q_1) = \pi_{Q_1} - \nu_{Q_1}$. The characteristic polynomial of Q_1 contains the eigenvalues of Q_1 as its roots. It is given by:

$$\lambda_{Q_1} = t^6 - (\frac{6}{15625} \cdot 640039)t^5 + (\frac{6^2}{15625^2} \cdot 8130390625)t^4.$$

Applying Descartes' rule of sign, the number of positve eigenvalues are computed to be $\pi_{Q_1} = Var(+,-,+) = 2$ and the number of negative eigenvalues can be verified to be $\nu_{Q_1} = Var(+,+,+) = 0$. Thus, $signature(Q_1) = \pi_{Q_1} - \nu_{Q_1} = 2 - 0 = 2$. The number of real solutions of F is 2 and therefore the consistency question can be answered with 'Yes'.

3.2 Improvements for Buchberger's Algorithm

This section presents several improvements for the basic version of Buchberger's algorithm as given by Algorithm 3.2.

Polynomials entering the Gröbner basis have been reduced completely with respect to all previous elements in the Gröbner basis using the NORMAL-FORM algorithm. But existing elements in the Gröbner basis are not tested for reducibility by the new entrants. This is called **interreduction** or **reverse reduction**. Buchberger states that these reverse reductions are essential for good performance, [Buc85]. It should be obvious, that any polynomial f which can be reduced to zero by a Gröbner basis G_1 can also be reduced to zero by a Gröbner basis G_2, if G_2 is obtained from G_1 by interreduction. This means, $0 \in NF_{G_1}(f) \Rightarrow 0 \in NF_{G_2}(f)$. The following version of Buchberger's algorithm includes the concept of reverse reduction in the FOR-loop. In case a polynomial h should be inserted into the non-complete Gröbner basis G, every g_j is checked whether it can be reduced by h. If $h' = g_j$, then g_j cannot be reduced by h. But, if the reduct h' is not equal to g_j, g_j is removed from G and all S-polynomials which are built with g_j, denoted by B_{help}, are removed from the set of all S-polynomials B. If h' is not equal to zero, all S-polynomials that can be constructed from h' and any element in G are added to B and h' is added to the Gröbner basis G. In case $h' = 0$, all g_j's in B_{help} are replaced by h and added to B as well as h is added to the Gröbner basis G.
In [CY94], the design and implementation of a parallel algorithm for computing Gröbner bases with interreduction on distributed memory multi-processors is presented. Therefore, the basic sequential version of Buchberger's algorithm is recasted as non-deterministic procedures, which consist of guarded commands, called rules.

Important efficiency improvements can be achieved by reducing the number of S-polynomials that have to be considered, since the polynomial reductions in the NORMAL-FORM algorithm are computationally the most expensive ones. In Buchberger's algorithm, the S-polynomials are checked for a zero remainder using the NORMAL-FORM algorithm with respect to a given set G. As already mentioned in Section 3.1, if G is a Gröbner basis, the order in which elements of G are selected for reduction is not relevant for the NORMAL-FORM algorithm. But in case G is not a Gröbner basis, the order in which elements of G are selected for reduction can effect the result of the remainder since the NORMAL-FORM algorithm transformes the set G into an ordered n-tuple. Thus, a stronger definition of the concept of reduction modulo G is needed.

Algorithm 3.7 GRÖBNER-BASIS-2(F)

Input: A finite set of interreduced polynomials $F \subseteq K[\mathbf{x}]$.
Output: A Gröbner Basis G for the ideal $Ideal(F)$.
$G := F$;
$B := \{S(f,g) \mid f,g \in F, f \neq g\}$; { $S(f,g)$ as defined in Corollary (3.6)}
while $B \neq \emptyset$ **do**
  $B := B \setminus S_i$; {Remove one S-polynomial from B}
  $h :=$ NORMAL-FORM(S_i, G);
  G-$tuple :=$ N-TUPLE(G); $\{g_j \in G\text{-}tuple\}$
  if $h \neq 0$ **then**
    for $j := 1$ to $|G|$ **do**
      $h' :=$ NORMAL-FORM$(g_j, \{h\})$;
      if $(h' \neq g_j)$ **then**
        $G := G \setminus g_j$;
        $B_{help} := \{S(g_j, g) \mid g \in G\} \cup \{S(g, g_j) \mid g \in G\}$;
        $B := B \setminus B_{help}$;
        if $(h' \neq 0)$ **then**
          $B := B \cup \{S(h', g) \mid g \in G\}$;
          $G := G \cup \{h'\}$;
        else
          $B := B \cup \{S(h, g) \mid S(g_j, g) \vee S(g, g_j) \in B_{help}\}$;
          $G := G \cup \{h\}$;
        end if
      end if
    end for
  end if
end while
return G ;

DEFINITION 3.14
Let $\underset{A}{>}$ *be a monomial ordering and let* $G = \{g_1, \cdots, g_q\}$ *be a subset in* $K[x_1, \cdots, x_n] = K[\mathbf{x}]$. *Then,* $f \in K[\mathbf{x}]$ *is* ***reduced to zero modulo G****, written* $f \longrightarrow_G 0$, *if* f *can be expressed in the form:*

$$f = a_1 g_1 + \cdots + a_t g_t, \tag{3.16}$$

such that whenever $a_i g_i \neq 0$, *the following holds:* $multideg(f) \geq multideg(a_i g_i)$.

LEMMA 3.19 *Let* $G = \{g_1, \cdots, g_q\}$ *be an ordered set of elements of* $K[\mathbf{x}]$ *and let* $f \in K[\mathbf{x}]$. *Then* $f \xrightarrow[*]{G} 0$ *implies* $f \longrightarrow_G 0$, *but not vice versa.*

EXAMPLE 3.13
Consider the following polynomial $f = xy^2 - x$ and the set $G = \{xy+1, y^2-1\}$. Applying the NORMAL-FORM algorithm results in $f \xrightarrow[*]{G} (-x-y) \neq 0$ since the first element of G is taken first for reduction: $f = xy^2 - x = y \cdot (xy+1) + 0 \cdot (y^2-1) + (-x-y)$. Using the reduction from Definition 3.14 instead, one yields $f \longrightarrow_G 0$ because f equals to $f = xy^2 - x = 0 \cdot (xy+1) + x \cdot (y^2-1) + 0$.

THEOREM 3.20 *A basis $G = \{g_1, \cdots, g_q\}$ for an ideal J is a Gröbner basis iff $S(g_i, g_j) \longrightarrow_G 0$ for all $i \neq j$.*

Theorem 3.20 provides a more general version of Definition 3.5 from Section 3.1. The next corollary describes certain S-polynomials which are guaranteed to reduce to zero.

COROLLARY 3.21 *Given a finite set $G \subset K[\mathbf{x}]$. If the head terms of $f, g \in G$ are relatively prime, i.e. $LCM(hterm(f), hterm(g)) = hterm(f) \cdot hterm(g)$, then the S-polynomial of f and g reduces to zero modulo G: $S(f, g) \longrightarrow_G 0$.*

Proof:
Assume that $hcoef(f) = hcoef(g) = 1$ and express f and g as $f = hterm(f) + p$ and $g = hterm(g) + q$, respectively. It follows from the definition of S-polynomials that:

$$\begin{aligned} S(f,g) &= hterm(g) \cdot f - hterm(f) \cdot g \\ &= (g - q) \cdot f - (f - p) \cdot g \\ &= g \cdot f - g \cdot f - f \cdot g + p \cdot g \\ &= p \cdot g - q \cdot f. \end{aligned}$$

In addition, one has to show that $multideg(S(f, g)) = max(multideg(pg), multideg(qf))$, in order to prove that $S(f, g) \longrightarrow_G 0$.

□

The next example illustrates the effectiveness of Corollary 3.21.

EXAMPLE 3.14
Let $G = \{yz + y, x^3 + y, z^4\}$ be a set of polynomials and let the monomial ordering be total degree ordering. Then, $S(x^3 + y, z^4) \longrightarrow_G 0$ applying Corollary 3.21, whereas using the NORMAL-FORM algorithm the following is computed: $S(x^3 + y, z^4) \xrightarrow[*]{G} y$. This is because the S-polynomial $S(x^3 + y, z^4) = yz^4$ is reduced by $yz - y$ to y: $yz^4 = (z^3 - z^2 - z - 1) \cdot (yz - y) + y$.

Therefore, in Buchberger's algorithm only those S-polynomials $S(f, g)$ have to be checked for reducibility to zero, where $hterm(f)$ and $hterm(g)$ are not relatively prime.

Another valuable improvement can be deduced from the role of S-polynomials in general. The name S-polynomial is an abbreviation for "syzygy polynomial", where the word syzygy comes from the greek word meaning "yoke". Thus, the properties of S-polynomials are studied in the following.

DEFINITION 3.15
*Let $F = \{f_1, \cdots, f_p\}$ be a set of polynomials. A **syzygy** on the set of head monomials $\{hmono(f_i) \mid 1 \leq i \leq p\}$ of F is a p-tuple of polynomials $S = (h_1, \cdots, h_p) \in (K[x_1, \cdots, x_n])^p$ such that $\sum_{i=1}^{p} h_i \cdot hmono(f_i) = 0$. The subset of $(K[x_1, \cdots, x_n])^p$ which consists of all syzygies on the head monomials of F is indicated by $S(F)$.*

The next example visualizes the definition of a syzygy.

EXAMPLE 3.15
Let $F = \{x, x^2+z, y+z\}$ be a set of polynomials in $K[x, y, z]$. Fix lexicographic ordering as a monomial ordering. Then $S = (-x + y, 1, -x) \in (K[x, y, z])^3$ is an example for a syzygy of $S(F)$ since $(-x+y) \cdot hmono(x)+1 \cdot hmono(x^2+z)+(-x) \cdot (y+z) = -x^2+yx+x^2-xy = 0$.

Now, the connection between syzygies and S-polynomials is developed. Let $\mathbf{e}_i$ be the vector $\mathbf{e}_i = (0, 0, \cdots, 1, 0, \cdots, 0) \in (K[\mathbf{x}])^p$, where 1 is in the i$^{\text{th}}$ place. Then, a syzygy S can also be expressed as a vector $\mathbf{S} = \sum_{i=1}^{p} h_i \cdot \mathbf{e}_i$. The syzygies that belong to the S-polynomials (also called syzygy-polynomials) from Buchberger's Gröbner basis characterization in Definition 3.5 are given by:

$$\mathbf{S}_{ij} = \frac{m}{hmono(f_i)}\mathbf{e}_i - \frac{m}{hmono(f_j)}\mathbf{e}_j, \tag{3.17}$$

where $m = LCM(hterm(f_i), hterm(f_j))$ for a set of two polynomials from F with $i < j$. $\mathbf{S}_{ij}$ is a syzygy on the head monomials of F. The set of syzygies $S(F)$ is closed under coordinate-wise sums and also under coordinate-wise multiplication with polynomials. $S(F)$ has a finite basis, such that every syzygy of $S(F)$ can be described as a linear combination of the basis syzygies.

For the next definition, recall the definition of the multidegree of a polynomial as given by Definition 2.39 from Section 2.2.

DEFINITION 3.16
An element $S \in S(F)$ is ***homogeneous of multidegree*** *α, where $\alpha \in \mathbb{Z}^n_{\geq 0}$, if S has the following structure:*

$$S = (c_1\mathbf{x}^{\alpha(1)}, \cdots, c_p\mathbf{x}^{\alpha(p)}),$$

with $c_i \in K$ and $\alpha(i) + multideg(f_i) = \alpha$ whenever $c_i \neq 0$.

LEMMA 3.22 *Every element of $S(F)$ can be written uniquely as a sum of homogeneous elements of $S(F)$.*

This lemma can be easily illustrated as follows: Suppose $S = (h_1, \cdots, h_p) \in S(F)$. According to the definition of a syzygy, it must be $\sum_{i=1}^{p} h_i hmono(f_i) = 0$. For a given $\alpha \in \mathbb{Z}^n_{\geq 0}$, $h_{i\alpha}$ defines the term of h_i (if any) such that $h_{i\alpha}hmono(f_i)$ has multidegree α. Then it follows that $\sum_{i=1}^{p} h_{i\alpha}hmono(f_i)$ also equals to 0 since the $h_{i\alpha}hmono(f_i)$ compose all terms of multidegree α in $\sum_{i=1}^{p} h_i hmono(f_i) = 0$. Thus, $S_\alpha = (h_{1\alpha}, \cdots, h_{p\alpha})$ is a homogeneous element of $S(F)$ of multidegree α and $S = \sum_\alpha S_\alpha$.

COROLLARY 3.23 *Given a set of polynomials $F = (f_1, \cdots, f_p)$, every syszgy $S \in S(F)$ can be written as*

$$S = \sum_{i<j} u_{ij} S_{ij},$$

where $u_{ij} \in K[x_1, \cdots, x_n]$ and the syzygy S_{ij} is defined as in Corollary (3.17).

It follows from Corollary 3.23, that the S_{ij}'s are a basis of all syzygies on the head monomials of F. This also illustrates that S-polynomials account for all possible cancellations of head monomials. Coming back to the issue of how to reduce the number of S-polynomials to be considered in Buchberger's algorithm, it turns out that some of the S_{ij}'s can be neglected because not always all of the S_{ij}'s are needed to create all syzygies in $S(F)$. Consider the following example.

EXAMPLE 3.16
Given the following set of polynomials $F = \{x^2y^2 + xy^2 - y, x^2y + yz\} \in K[x, y, z]$ and let the monomial ordering be lexicographic ordering. Then, one can compute that the three syzygies belonging to the S-polynomials are equal to:

$$S_{12} = (1, -x, 0), \quad S_{13} = (1, 0, -y) \text{ and } \quad S_{23} = (0, x, -y).$$

For example, S_{23} is redundant since it can be expressed as a linear combination of S_{12} and S_{13}: $S_{23} = S_{13} - S_{12}$ and thus can be eliminated from the basis. Therefore, the set $\{S_{12}, S_{13}\}$ forms a basis for the syzygies. (It is also possible to express S_{12} or S_{13} as a linear combination of the other two, respectively.)

The following theorem presents a refined definition of a Gröbner basis compared to Definition 3.5 from Buchberger's Characterization. If the basis $\{S_{ij}\}$ is used for the syzygies $S(G)$, then the polynomials $S_{ij} \cdot G$ are precisely the S-polynomials $S(g_i, g_j)$.

THEOREM 3.24 *A basis $G = \{g_1, \cdots, g_q\}$ for an ideal $J = Ideal(f_1, \cdots, f_p)$ is a Gröbner basis iff for every element $S_j = (h_{j1}, \cdots, h_{jt})$ in a homogeneous basis $S = \{S_1, \cdots, S_s\}$ for the syzygies $S(G)$ it holds that:*

$$S_j \cdot G = \sum_{i=1}^{t} h_{ji} \cdot g_i \longrightarrow_G 0. \tag{3.18}$$

Observe that if the basis $\{S_{ij}\}$ is used for the syzygies $S(G)$, then the polynomials $S_{ij} \cdot G$ to be tested compose the special case of the S-polynomials $S(g_i, g_j)$.

COROLLARY 3.25 *Given $G = \{g_1, \cdots, g_q\}$, suppose a subset $S \subset \{S_{ij} \,|\, 1 \leq i < j \leq t\}$ which is a basis of $S(G)$. Assume distinct $g_i, g_j, g_k \in G$ such that $hmono(g_k)$ divides $LCM(hmono(g_i), hmono(g_j))$. If $S_{ik}, S_{jk} \in S$, then $S \setminus \{S_{ij}\}$ is also a basis of $S(G)$. (Note that if $i > j$ then $S_{ij} = S_{ji}$.)*

Proof:
Assume that $i < j < k$. Let $m_{ij} = LCM(hterm(g_i), hterm(g_j))$ and let m_{ik} and m_{jk} be defined in the same way. If there are distinct $g_i, g_j, g_k \in G$ such that $hmono(g_k)$ divides $LCM(hmono(g_i), hmono(g_j))$, then this implies that m_{ik} and m_{jk} also both divide m_{ij}. The reason for this is because $hmono(g_k)$ is a factor of m_{ik} and m_{jk} since the $hcoef(g_k)$ can be neglected here. In order to show, that $S \setminus \{S_{ij}\}$ is also a basis of $S(G)$, one has to prove that S_{ij} is a linear combination of S_{ik} and S_{jk}. Using Corollary (3.17) for S_{ik} and S_{jk}, it follows that:

$$\begin{aligned} S_{ij} &= \frac{m_{ij}}{m_{ik}} S_{ik} - \frac{m_{ij}}{m_{jk}} S_{jk} \\ &= \frac{m_{ij}}{m_{ik}} \left(\frac{m_{ik}}{hmomo(f_i)}\mathbf{e}_i - \frac{m_{ik}}{hmono(f_k)}\mathbf{e}_k\right) - \frac{m_{ij}}{m_{jk}} \left(\frac{m_{jk}}{hmomo(f_j)}\mathbf{e}_j - \frac{m_{jk}}{hmono(f_k)}\mathbf{e}_k\right) \\ &= \frac{m_{ij}}{hmomo(f_i)}\mathbf{e}_i - \frac{m_{ij}}{hmono(f_j)}\mathbf{e}_j \end{aligned}$$

□

Corollary 3.25 states which elements of the basis $\{S_{ij}\}$ can be omitted. Therefore, ordered pairs (i, j) are used to denote the syzygies and the corresponding S-polynomials. The following version of Buchberger's algorithm, Algorithm 3.8, includes the improvements which

can be achieved by Corollary 3.21 and Corollary 3.25. The algorithm starts with all S_{ij}'s (the set of all ordered pairs B), which are known to be a basis of $S(G)$ from Corollary 3.23. It then takes then only those syzygies and computes the remainder of corresponding S-polynomials with the NORMAL-FORM algorithm, for which neither Corollary 3.21 nor Corollary 3.25 apply. Corallary 3.25 is implemented in the CRITERION algorithm which is presented in Algorithm 3.9. In the CRITERION algorithm, the pair $[i, j]$ is defined as:

$$[i,j] = \begin{cases} (i,j) & \text{if } i < j \\ (j,i) & \text{if } i > j, \end{cases} \tag{3.19}$$

because sometimes for a given $i \neq j$ one does know which one is larger. If the pairs $[i, k]$ and $[j, k]$ are not in the set of ordered pairs B, then they must have been removed earlier and thus S_{ik} and S_{jk} have already been computed to be in the basis S. Then, S_{ij} does not have to be considered since $S \setminus \{S_{ij}\}$ is also a basis of $S(G)$.

Algorithm 3.8 GRÖBNER-BASIS-3(F)

Input: A finite set of polynomials $F \subseteq K[\mathbf{x}]$.
Output: A Gröbner Basis G for the ideal $Ideal(F)$.
$G := F$; $t := |G|$;
G-tuple := N-TUPLE(G); $\{f_i, f_j \in$ *G-tuple*$\}$
$B := \{(i,j) \mid 1 \leq i < j \leq t\}$; {ordered pairs (i, j) to denote the syzygies}
while $B \neq \emptyset$ **do**
  $B := B \setminus (i,j)$; {Remove one "syzygy" from B}
  if $(LCM(hmono(f_i), hmono(f_j)) \neq hmono(f_i)hmono(f_j))$
  and $(\neg CRETERION(f_i, f_j, F, B)$ **then**
    $S := S(f_i, f_j)$**;**
    $h :=$ **NORMAL-FORM**(S, G)**;**
    if $h \neq 0$ **then**
      $t := t + 1$**;** $f_t := h$**;**
      $G := G \cup \{f_t\}$**;**
      G-tuple := **N-TUPLE**(G)**;** **{**$f_i, f_j \in$ *G-tuple***}**
      $B := B \cup \{(i,t) \mid 1 \leq i \leq t-1\}$**;**
    end if
  end if
end while
return G ;

For a detailed proof of this algorithm, please refer to [CLO96].

3.3 Assessments of Gröbner Basis Methods

The basic Gröbner basis method can decide the existential problem for a system of real non-linear multivariate polynomials in any algebraically closed field by applying the ideal membership question for the following class of formulas:

$$(quantifiers)\ (arbitrary\ boolean\ combination\ of\ polynomial\ equations), \tag{3.20}$$

Algorithm 3.9 CRITERION(f_i, f_j, F, B)

Input: Two polynomials f_i and f_j from the set F of polynomials and the set B of ordered pairs to denote the syzygies and corresponding S-polynomials.
Output: TRUE or FALSE.
$output := FALSE$; $k := 1$;
$m_{ij} := LCM(hmono(f_i), hmono(f_j))$;
while $(k \neq |F| + 1)$ **and** $(\neg output)$ **do**
  if $(k \neq i)$ **and** $(k \neq j)$ **then**
    if $([i,k] \notin B)$ **and** $([j,k] \notin B)$ **and** $(hmono(f_k)$ **divides** $m_{ij})$ **then**
      $output := TRUE$**;**
    else
      $k := k + 1$**;**
    end if
  end if
end while
return *output* **;**

where all quantifieres are either $\exists$ or $\forall$. Furthermore, the formula must be closed, i.e. no free variables may occur. Any polynomial inequation $f \neq 0$ can be transformed to a polynomial equation $u \cdot f = 1$, where u is a new variable (slack variable). Inequalities containing ordering relations such as $<, >, \leq$ and $\geq$ cannot be handled with this method. It is not completely possible to use the basic Gröbner basis method in a field K which is not algebraically closed because it then only works in one direction. In this case, the method provides the results for the algebraic closure of K. This applies also to the interesting field of the reals and the complex' as their algebraic closure. Posing the decision problem for a system of real non-linear multivariate polynomials, one can only conclude that it has no common real zeros if and only if it has no common complex zeros. (In this case, the Gröbner basis contains a constant polynomial.) On the other hand, if the system has common complex zeros, one cannot deduce if it has also real ones since $\mathbb{R}$ is not algebraically closed.

In the application field of geometrical theorem proving, the basic Gröbner basis method can be used to verify the following class of geometrical theorems:

$$\forall \mathbf{x} \in \mathbb{R}((h_1 = 0 \wedge \cdots \wedge h_r = 0 \wedge d_1 \neq 0 \wedge \cdots \wedge d_t \neq 0) \Longrightarrow c = 0), \tag{3.21}$$

where $h_1, \cdots, h_r$ denote the hypotheses, $d_1, \cdots, d_t$ indicates the degenerate situations and c represents the conclusion. The hypotheses, degenerate situations, and the conclusion have to be expressed as polynomial expressions. The geometrical theorem is proved by moving the problem from $\mathbb{R}$ to its algebraic closure $\mathbb{C}$ and then by solving the radical membership question for c, i.e $c \in Ideal(h_1, \cdots, h_r, u_1 \cdot d_1 - 1, \cdots, u_t \cdot d_t - 1, v \cdot c - 1)$. Again, any polynomial inequation can be transformed to a polynomial equation using slack variables. Inequalities involving ordering relations such as $<, >, \leq$ and $\geq$ cannot be treated with this method. Geometrical theorems can only be confirmed but not refuted for the same reason again, that $\mathbb{R}$ is not algebraically closed and the problem has to be shifted to its algebraic closure $\mathbb{C}$ in order to apply Gröbner bases. Therefore, if a geometrical theorem holds over $\mathbb{C}$ then it also holds over $\mathbb{R}$. The reverse cannot be implied in general.

Gröbner bases can be used to solve a system of real non-linear multivariate polynomials if the ideal generated by these polynomials is zero-dimensional, which means that the system has finitely many solutions. Thus, the enumeration question as defined in the introduction can be answered. Therefore, two methods have been presented: the elimination method and the univariate-polynomial method. These two methods can be improved using algebraic numbers to represent the exact solutions. If it is possible to answer the enumeration question, one can also easily answer the existential and the counting question.

The consistency problem for a system of real non-linear multivariate polynomial equations and inequations can be solved by a method which counts the real zeros under one side condition. However, this approach is also constrained to zero dimensional ideals. Given a set of polynomials $F = \{f_1, \cdots, f_p\}$ and a side condition h in $K[x_1, \cdots, x_n]$, this method counts the zeros in $\mathbb{R}^n$ for which $f_i(x_1, \cdots, x_n) = 0$ for all $1 \leq i \leq p$ and $h(x_1, \cdots, x_n)\, \rho\, 0$, where $\rho \in \{<, =, >\}$. Inequalities are transformed to equations as explained before. Inequalities including ordering relations such as $<, >, \leq$ and $\geq$ can occur only once in this basic version and that is in the side condition. Having more than one inequality, one has to apply the extended version of this approach, which allows finitely many side conditions. Given a set of polynomials $F = \{f_1, \cdots, f_p\}$ and finitely many side conditions $(h_1, \cdots, h_s)$ in $K[x_1, \cdots, x_n]$, the extended method counts the zeros in $\mathbb{R}^n$ for which $f_i(x_1, \cdots, x_n) = 0$ for all $1 \leq i \leq p$ and $h_j(x_1, \cdots, x_n)\, \rho\, 0$ for all $1 \leq j \leq s$ and where $\rho \in \{<, =, >\}$. Thus, this method can solve the quantifier elimination problem for the elementary theory of the reals for zero-dimensional ideals. It can handle the following type of formulas:

$$\exists x_1 \cdots x_k (\bigwedge_{i=1}^{p} f_i(x_1, \cdots, x_n) = 0 \wedge \bigwedge_{i=1}^{s} h_i(x_1, \cdots, x_n)\, \rho\, 0), \tag{3.22}$$

with $1 \leq k \leq n$, $f_i, h_i \in \mathbb{Q}[x_1, \cdots, x_n]$ and $\rho \in \{<, =, >\}$. Formulas of the form $\forall x(\varphi)$ for any quantifier-free formula φ can be managed by inverting them to their equivalent form $\neg\exists x(\neg\varphi)$).

The complexity of Buchberger's algorithm is still an active area of research. One can still generate examples of ideals for which the computation of a Gröbner basis exceeds time or storage limits. The reason for this is that the total degree of intermediate polynomials can become very huge. Moreover, the rational coefficients of the Gröbner basis generators can be quite complicated, even though the coefficients of the input polynomials which create the ideal are small integers. In literature, several upper worst case bounds on the degrees of intermediate polynomials have been depicted. However, time and storage capacities required by the algorithm seem to be less extreme in the average case. Additionally, computing geometric information is less complex.

In general, computing Gröbner Bases with respect to lexicographic ordering needs more computation time than using total degree ordering or reverse lexicographic ordering. Therefore, it might be important to convert between Gröbner bases with respect to different admissible monomial orderings, as for example in the application field of implicitizations for surfaces in polynomial parametric form. In [Tra00], Tran presents a fast algorithm that computes for a given reduced Gröbner basis of an ideal $J \subset K[x_1, \cdots, x_n]$ with respect to an admissible ordering a reduced Gröbner basis of J with respect to another admissible ordering without applying Buchberger's algorithm. The precedence of the variables involved also seems to play an important role concerning complexity issues.

Considering a decision problem in an incremental situation as described in the introduction, one can observe that the Gröbner basis method is applicable for it. A version of Buchberger's algorithm is requested that computes a new Gröbner basis from a given Gröbner basis and an additional polynomial. This version does not have to consider S-polynomials which are constructed from elements of the old Gröbner basis G because all of those S-polynomials are reducible by G as stated in Definition 3.5. Thus, only those S-polynomials generated from the additional polynomial and a generator of G need to be treated. The Algorithm 3.10 presents this version of Buchberger's algorithm.

Algorithm 3.10 GRÖBNER-BASIS-4(G, f)

Input: A Gröbner basis $G \subseteq K[\mathbf{x}]$ and an additional polynomial f.
Output: A new Gröbner basis G_{new} for the ideal $Ideal(G, f)$.
$G_{new} := G$;
$B := \{S(f, g) \mid g \in G\}$; { $S(f, g)$ as defined in Corollary (3.6)}
while $B \neq \emptyset$ **do**
  $B := B \setminus S_i$; {Remove one S-polynomial from B}
  $h :=$ NORMAL-FORM(S_i, G_{new});
  if $h \neq 0$ **then**
    $B := B \cup \{S(g, h) \mid g \in G_{new}\}$;
    $G_{new} := G_{new} \cup \{h\}$;
  end if
end while
return G_{new} ;

While traversing the tree-structure of polynomials, the Gröbner basis at each node and leaf has to be saved for the purpose of backtracking. Backtracking needs to be performed in case a new Gröbner basis leads to an inconsistent set of polynomials.

Chapter 4

Solving the Decision Problem using Cylindrical Algebraic Decomposition

In this chapter, Collins' constructive quantifier elimination method of finding a real solution to a system of non-linear multivariate polynomial equations and inequations, a real polynomial system, is analyzed to solve the decision problem with respect to the application background of automated reasoning with description logics. The Cylindrical Algebraic Decomposition (CAD) algorithm was discovered by Collins in 1973, [Col98], as a basic tool for his effective method for quantifier elmination in real closed fields. This chapter starts with a presentation of Collins' CAD algorithm. Real algebraic numbers are the basic objects of the CAD algorithm. Therefore, the representation of real algebraic numbers as well as the calculation with real algebraic numbers are considered in a separate section. The chapter concludes with assessments of Collin's method.

4.1 Collins' CAD Algorithm

This section is mainly based on [Jir95] and [ACM98]. First of all, the main algebraic concepts for cylindrical algebraic decompositon are introduced followed by an introductory example in order to explain the general outline of Collins' Cylindrical Algebraic Decomposition (CAD). In general, a cylindrical algebraic decomposition of $\mathbb{R}^n$ partitions $\mathbb{R}^n$ into connected subsets, called cells, on which each of the polynomials has constant sign.

DEFINITION 4.1
A ***region*** $\mathcal{R}$ *is a connected subset of* $\mathbb{R}^n$.
The set $\mathcal{Z}(\mathcal{R}) = \mathcal{R} \times \mathbb{R} = \{(\alpha, x) \mid \alpha \in \mathcal{R}, x \in \mathbb{R}\}$ *is called a* ***cylinder*** *over* $\mathcal{R}$.
Let f, f_1, f_2 *be continuous, real-valued functions on* $\mathcal{R}$. *A* ***f-section*** *of* $\mathcal{Z}(\mathcal{R})$ *is the set* $\{\alpha, f(\alpha)) \mid \alpha \in \mathcal{R}\}$ *and a* (f_1, f_2)***-sector*** *of* $\mathcal{Z}(\mathcal{R})$ *is the set* $\{(\alpha, \beta) \mid \alpha \in \mathcal{R}, f_1(\alpha) < \beta < f_2(\alpha)\}$.

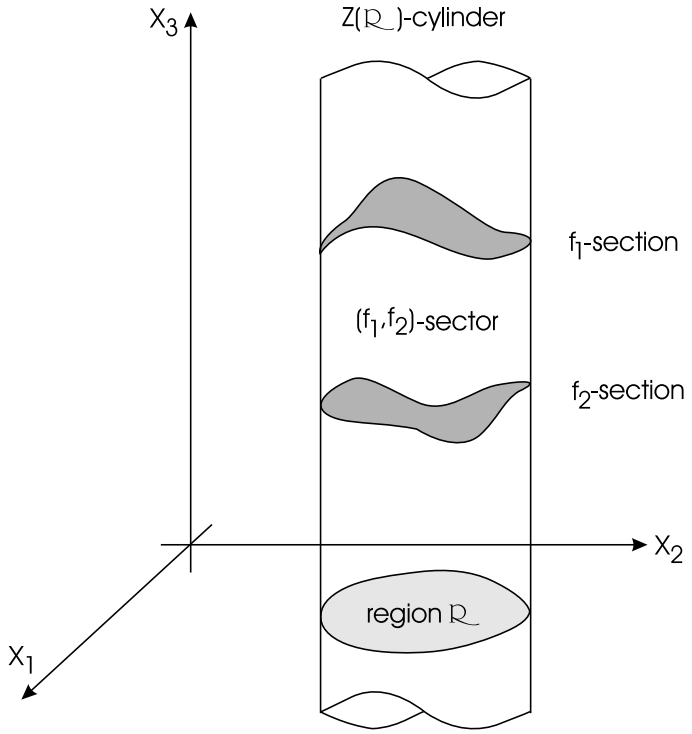

Figure 4.1: Geometrical interpretations of Definition 4.1(Figure taken from [Jir95]).

DEFINITION 4.2
Let $X \subseteq \mathbb{R}^n$. A ***decomposition of X*** *is a finite collection of disjoint regions (components) whose union is X:*

$$X = \bigcup_{i=1}^{k} X_i, \quad X_i \cap X_j = \emptyset, \quad i \neq j$$

DEFINITION 4.3
A ***stack over $\mathcal{R}$*** *is a decomposition of a cylinder $\mathcal{Z}(\mathcal{R})$ which consists of f_i -sections and (f_i, f_{i+1}) -sectors, where $f_0 < \ldots < f_{k+1}$ for all $x \in \mathcal{R}$ and $f_0 = -\infty, f_{k+1} = +\infty$.*

DEFINITION 4.4
A decomposition $\mathcal{D}$ of $\mathbb{R}^n$ is ***cylindrical*** *if either $n = 1$ and $\mathcal{D}$ is a partition of $\mathbb{R}^1$ into a finite set of numbers and a finite set of open intervals bounded by these numbers, $-\infty$ and $+\infty$ or $n > 1$ and $\mathcal{D}^{n-1} = X_1 \cap \ldots \cap X_k$ is a cylindrical decomposition of $\mathbb{R}^{n-1}$ and over each X_i there is a stack which is a subset of $\mathcal{D}$.*

DEFINITION 4.5
A decomposition is ***algebraic*** *if each of its components is a semi-algebraic set.*

DEFINITION 4.6
A ***cylindrical algebraic decomposition (CAD)*** *of $\mathbb{R}^n$ is a decomposition which is both cylindrical and algebraic. The components of a CAD are called cells.*

DEFINITION 4.7
Let $X \subseteq \mathbb{R}^n$ and $f \in K[x_1, \ldots, x_n]$. The polynomial f is ***invariant on X*** *or one can equally say X is* ***f-invariant*** *iff one of the following conditions holds for the corresponding polynomial function $f(x)$:*

$$(i)\ \forall x \in X : f(x) > 0$$

$$(ii)\ \forall x \in X : f(x) = 0$$

$$(iii)\ \forall x \in X : f(x) < 0$$

The set $\mathcal{F} = f_1, \ldots, f_r \subset K[x_1, \ldots, x_n]$ of polynomials is ***invariant on X*** *or X is* ***$\mathcal{F}$ -invariant*** *iff each f_i is invariant on X.*

DEFINITION 4.8
Let f be a polynomial $\in \mathbb{Z}[x_1, \ldots, x_n]$, $n \geq 1$ and $\mathcal{R}$ be a region in $\mathbb{R}^{n-1}$. Then, f is called ***delineable*** *on $\mathcal{R}$ if the portion of $ZERO(f)$ lying in $\mathcal{Z}(\mathcal{R})$ consists of s disjoint sections of $\mathcal{Z}(\mathcal{R})$, for some $s \geq 0$.*

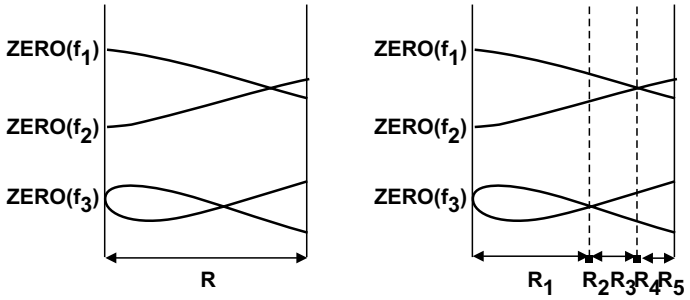

Figure 4.2: Illustration of the delineability feature (Figure taken from [ACM98]).

Figure 4.2 illustrates Definition 4.8. On the left hand side of the figure, region $\mathcal{R}$ is shown with the zero sets of three bivariate polynominals f_1, f_2 and f_3. f_1 and f_2 are delineable on $\mathcal{R}$. The delineability condition does not hold for f_3 because its zero set is self-crossing. The right hand drawing presents a partition of region $\mathcal{R}$ into five smaller regions $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_5$, such that all three polynomials are delineable on all regions $\mathcal{R}_1, \mathcal{R}_2, \cdots, \mathcal{R}_5$. If a set of polynomials $\mathcal{F} \in K[x_1, \ldots, x_{n-1}][x_n]$ is delineable on a region $\mathcal{R}$, one can rise a stack over $\mathcal{R}$ which is obviously $\mathcal{F}$-invariant. The delineability property implies that the total number of complex roots (counting multiplicity), the number of distinct complex roots, and the total number of common complex roots (counting multiplicity) of every $f_i \in \mathcal{F}$ remains invariant as α varies over the coefficient field $K[x_1, \ldots, x_{n-1}]$. A proof can be found in [Mis93].

The following example is presented as an introductory example.

EXAMPLE 4.1
Given the real polynomial system:

$$x_1^2 + x_2^2 - 1 < 0$$
$$x_1^3 - x_2^2 = 0.$$

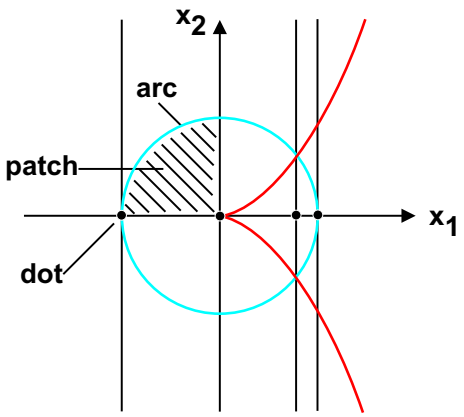

Figure 4.3: Zero sets of the polynomial system.

The task is to determine, whether the semi-algebraic set of the given real polynomial system is empty or not, considering only real solutions. The defining polynomials are projected onto the x_1-axis such that the roots of the received univariate polynomials represent their singularities and vertical tangent points. Singularities are points like selfcrossings, isolated points and cusps. Points at which their tangent is vertical are called vertical tangent points. With this projection, $\mathbb{R}^1$ is decomposed into points and several open intervals. The open intervals are located inbetween two adjacent points as well as before and after the first and last point, respectively. "Drawing" vertical lines over each point, one receives vertical stripes over each open interval. The vertical lines consist of finitely many disjoint **dots** and **straight arcs**, whereas the vertical stripes contain finitely many **arcs** and **patches** of space. All these dots, arcs and patches constitute the cylindrical algebraic decomposition, for short also called CAD, of the plane in this example. Analyzing the sign of the polynomials in each element of the decomposition yields the solution.

All these open and closed intervals are regions according to Definition 4.1. Over all open and closed intervals, stacks are constructed, such that the f_i -sections and (f_i, f_{i+1}) -sectors do not cross each other. The projection has to be performed in such a way that it is guaranteed to be able to build a stack over each region in each dimension.

Coming back to Example 4.1, the question is to determine whether the real polynomial system has a common real zero set. In Figure 4.3, the zero sets of the two defining polynomials are shown. The zero set of the first polynomial equation consists of each point inside the drawn circle excluding the points on the circle itself, whereas the zero set for the second one contains each point on the two hyperbola branches. Obviously, the common real zero set of the real polynomial system is defined by all points on the two hyperbola branches from the origin of the coordinate system up to the circle. Using the CAD algorithm from Collins', one simply has to perform one single projection from $\mathbb{R}^2$to $\mathbb{R}^1$ since this example is only two-dimensional. The zeros of this projection define the singularities andvertical tangent points, which are represented by the 4 black dots on the x_1-axis. From left to right, the first dot is a vertical tangent point, the second one denotes a cusp, the third stands for the crossing point of the two zero sets of the input polynomials and the latter point is again a vertical tangent point. In a second step the x_1-axis is decomposed into several intervals according to the previous projection. The zero

points of the projection constitute to closed point intervals and at the same time depict the borders for the open intervals. The open intervals lie inbetween two adjacent zero points and before and after the first and last zero point up to negative or positive infinity, respectively. As a whole, there are 9 intervals. A sample point for each interval is selected. Finally, the decomposition of $\mathbb{R}^1$ has to be extended back to $\mathbb{R}^2$. This means that a stack has to be constructed over each region of the decomposition of $\mathbb{R}^1$. Therefore, each of the sample points is inserted into the set of polynomials of the next higher dimension, which is in this case $\mathbb{R}^2$. One receives again polynomials in one variable whose zero sets can be calculated. These zero points are the basis for the decomposition of the vertical line or the vertical stripe over the chosen cell of $\mathbb{R}^1$. Sample points are also determined for the intervals. Together with the decomposition of $\mathbb{R}^1$, this results in a decomposition of the two-dimensional space. A sample point of a two-dimensional cell consists in the first coordinate of a sample point from the decomposition of the x_1-axis and in the second coordinate of a sample point from the decomposition in the second variable x_2 over the first coordinate. The signs of the defining polynomials are then evaluated for all the cells of this two-dimensional cylindrical algebraic decomposition. This is simply done by inserting the sample point of every cell into the polynomial equations. If any cell can be found in this example for which the sign of the first polynomial is negative and for which the sign of the second one is equal to zero, then the real polynomial system has a real solution.

The basic outline of Collins' CAD algorithm in $\mathbb{R}^n$ is as follows:

- Successive projection of the set of polynomials into $\mathbb{R}^{n-1}, \cdots, \mathbb{R}^1$. The zeros of each set depict the "signature" of the singularities and vertical tangent points of the next higher dimensional space.
- Computation of the roots of the polynomials in $\mathbb{R}^1$, decomposition of $\mathbb{R}^1$ into points and open intervals and determination of sample points for each interval.
- Extension of the decomposition of $\mathbb{R}^1$ to $\mathbb{R}^2, \cdots, \mathbb{R}^n$.

The important question is how to define a projection such that a stack can always be constructed over each region in every n-dimensional space. First, the desired properties of such a projection operator have to be defined. In the following, the projection operator is called PROJECTION. The desired properties are:

- Any PROJECTION($\mathcal{F}$)-invariant cylindrical algebraic decomposition of $\mathbb{R}^{n-1}$ is induced by some $\mathcal{F}$-invarant cylindrical algebraic decomposition of $\mathbb{R}^n$, where $\mathcal{F}$ is a set of input polynomials.
- For any PROJECTION($\mathcal{F}$)-invariant region $\mathcal{R}$ the following conditions hold:
  (i) Each $f_i \in \mathcal{F}$ is either delineable or identical zero on $\mathcal{R}$.
  (ii) The sections of $\mathcal{Z}(\mathcal{R})$ belonging to different f_i, f_j are either disjoint or identical.

Now, the first part of the projection operator is developed. The next theorem provides the basis for this.

THEOREM 4.1 *Let $f \in K[x_1, \ldots, x_{n-1}][x_n]$ and let $\mathcal{R}$ be a region in $\mathbb{R}^{n-1}$. f_α denotes $f(\alpha, x_n)$, where $\alpha \in \mathbb{R}^{n-1}$. Suppose that $deg(f_\alpha)$ is constant and nonnegative for $\alpha \in \mathcal{R}$. If the least k such that $psc_k(f_\alpha, f'_\alpha) \neq 0$ is constant for $\alpha \in \mathcal{R}$, then f is delineable on $\mathcal{R}$.*

DEFINITION 4.9 ($PROJECTION_1(\mathcal{F})$)
Let $\mathcal{F} = \{f_1, \cdots, f_p\}, p \geq 1$ be a set of polynomials in $K[x_1, \ldots, x_{n-1}][x_n]$. The first part of the projection operator is denoted by $PROJECTION_1$ such that $PROJECTION_1(\mathcal{F}) \subset K[x_1, \ldots, x_{n-2}][x_{n-1}]$. For each i, $1 \leq i \leq p$, let $R_i = RED(f_i)$. Then $PROJECTION_1(\mathcal{F})$ is defined as follows:

$$PROJECTION_1(\mathcal{F}) = \bigcup_{i=1}^{p} \bigcup_{G_i \in R_i} (\{HEADCOEF(G_i)\} \cup PSC(G_i, G_i')). \tag{4.1}$$

Given a $PROJECTION_1(\mathcal{F})$-invariant region $\mathcal{R}$, Theorem 4.1 implies for each $f_i \in \mathcal{F}$ that $deg((f_i)_\alpha)$ is constant $\forall \alpha \in \mathcal{R}$. Consider Example 2.9 from the Section 2.2 again. The degree of $f(x, y, z)$ changes for different regions $\mathcal{R}$ in $K[x, y]$. If $\mathcal{R}$ is the region disjoint from the unit circle, then $f(x, y, z)$ has degree 3 $\forall \alpha \in \mathcal{R}$. $f(x, y, z)$ equals the zero polynomial, in case region $\mathcal{R}$ contains only the point (1,0) from the unit circle. If $\mathcal{R}$ is a subset of the unit circle, except the point (1,0), then $f(x, y, z)$ is of degree 2. All these different cases are taken into consideration by the projection operator since all the head coefficients of each element of the set of reducta of $f(x, y, z)$ are included in the set of polynomials for the projection. If for every $f_i \in \mathcal{F}$ holds that $deg((f_i)_\alpha)$ is constant $\forall \alpha \in \mathcal{R}$, it also means that the total number of complex roots of f_i with counting multiplicity remains invariant as α varies over $\mathcal{R}$. The invariance of the principle subresultant coefficient set of any element of the set of reducta of $f(x, y, z)$ and its derivative, denoted by $PSC(G_i, G_i')$ in Equation (4.1), provides that the greatest common divisor of each polynomial and its derivative has constant degree according to Lemma 2.9. Applying Lemma 2.8 and using the invariance of $deg((f_i)_\alpha)$, denoted by $HEADCOEF(G_i)$ in equation 4.1, it can be derived that the number of distinct complex zeros of each polynomial in $\mathcal{F}$ is constant, as α varies over $\mathcal{R}$. For a detailed proof, please refer to [Col98] and [Mis93].

Now, the main theorem for the second part of the projection operator is formulated as well as the definition of it is presented.

THEOREM 4.2 *Let $\mathcal{F} \in K[x_1, \ldots, x_{n-1}][x_n]$ and let $\mathcal{R}$ be a region in $\mathbb{R}^{n-1}$. Suppose that for every $f \in \mathcal{F}$, the hypotheses of Theorem 4.1 are satisfied. Suppose also that for every $f, g \in \mathcal{F}, f \neq g$, the least k such that $psc_k(f_\alpha, g_\alpha) \neq 0$ is constant for $\alpha \in \mathcal{R}$. Then every $f \in \mathcal{F}$ is delineable on $\mathcal{R}$ and for every $f, g \in \mathcal{F}$, any f-section and any g-section of $\mathcal{Z}(\mathcal{R})$ are either disjoint or identical.*

DEFINITION 4.10 ($PROJECTION_2(\mathcal{F})$)
Let $\mathcal{F} = \{f_1, \cdots, f_p\}, p \geq 1$, be a set of polynomials in $K[x_1, \ldots, x_{n-1}][x_n]$. For each i, $1 \leq i \leq p$, let $R_i = RED(f_i)$. Then, the second part of the projection operator, denoted by $PROJECTION_2$, is defined as follows:

$$PROJECTION_2(\mathcal{F}) = \bigcup_{1 \leq i < j \leq p} \bigcup_{G_i \in R_i \,\&\, G_j \in R_j} PSC(G_i, G_j). \tag{4.2}$$

The invariance of $PROJECTION_2(\mathcal{F})$ in addition to the fact that $deg((f_i)_\alpha)$ as well as $deg((f_j)_\alpha)$ are constant according to the definition of $PROJECTION_1(\mathcal{F})$ in Corollary (4.1) causes the total number of common complex roots of f_i and f_j for every $1 \leq i < j \leq p$ to remain invariant as α varies over $\mathcal{R}$. This is achieved in Corollary (4.2)

by including the principle subresultant coefficient set of any element of the set of reducta of f_i and f_j respectively, denoted by $PSC(G_i, G_j)$. The invariance of the lattersupplies again that the greatest common divisor of each polynomial of the set of reducta of f_i and f_j has constant degree according to Lemma 2.9. Finally, it can be derived that the number of common complexzeros of f_i and f_j stays invariant as α varies over $\mathcal{R}$ using Lemma 2.7 and applying the invariance of $deg((f_i)_\alpha)$. The proof is omitted here. [Col98] and [Mis93] provide a more detailed explanation.

The complete projection operator, denoted by $PROJECTION(\mathcal{F})$, is then defined to be the union of the first and the second part of the projection operator as given by Corollary (4.1) and Corollary (4.2) respectively:

$$PROJECTION(\mathcal{F}) = PROJECTION_1(\mathcal{F}) + PROJECTION_2(\mathcal{F}). \tag{4.3}$$

With the definition of the projection operator, the algorithm for cylindrical algebraic decomposition (CAD algorithm) can now be explained in detail. The CAD algorithm is used to answer the question whether a real polynomial system has a common real solution. It takes a set $\mathcal{F}$ of multivariate polynomials as input and provides an $\mathcal{F}$-invariant cylindrical algebraic decomposition of $\mathbb{R}^n$ as output. As already stated above, the CAD algorithm consists of three parts: the projection phase, the base phase and the extension phase.

The **projection phase** takes the set $\mathcal{F} = f_1, \cdots, f_p \subset \mathbb{R}[x_1, \cdots, x_n]$ of input polynomials and applies the projection operator recursively $n-1$ times. This results in successive projections of the set of input polynomials into $\mathbb{R}^{n-1}, \cdots, \mathbb{R}^1$. n denotes the number of variables. In each step, a new set of polynomials is constructed and the number of variables is decreased by one. The zero set of each projection set of polynomials depicts "significant" points of the zero sets of the preceeding higher dimensional projection set.

Let $\mathcal{F} = f_1, \cdots, f_p$, $PROJECTION^0(\mathcal{F}) = \mathcal{F}$ and $PROJECTION^j(\mathcal{F}) = PROJECTION(PROJECTION^{j-1}(\mathcal{F}))$ for $1 \leq j \leq n-1$. Then, the following projection sets are obtained:

$$\begin{aligned}
\mathcal{F}^n &= \mathcal{F} \subset \mathbb{R}[x_1, \cdots, x_n] \\
\mathcal{F}^{n-1} &= PROJECTION^1(\mathcal{F}) \subset \mathbb{R}[x_1, \cdots, x_{n-1}] \\
\mathcal{F}^{n-2} &= PROJECTION^2(\mathcal{F}) \subset \mathbb{R}[x_1, \cdots, x_{n-2}] \\
&\vdots \\
\mathcal{F}^1 &= PROJECTION^{n-1}(\mathcal{F}) \subset \mathbb{R}[x_1]
\end{aligned}$$

In the **base phase**, the real roots of the univariate polynomials of the last projection set $PROJECTION^{n-1}(\mathcal{F})$ are isolated. They define a sign invariant decomposition of $\mathbb{R}^1$ into points and open intervals. The points or zeros can also be considered as closed intervals. For each closed and open interval a sample point of the decomposition of $\mathbb{R}^1$ is chosen. This means the zeros themselves are selected as well as one intermediate point for each open interval.

Let $\mathcal{F}^j = f_{j,1}, \cdots, f_{j,p}$ and take $j = 1$ here. (The j in $\mathcal{F}^j$ corresponds to the dimension of polynomials in the projection sets.) Then the zeros α_i of all polynomials $f_{j,i}$ in $\mathcal{F}^j$ are enumerated as follows:

$$-\infty < \alpha_1 < \alpha_2 < \cdots < \alpha_s < +\infty.$$

The base phase provides the real root isolation of these enumerated zeros such that

$$\alpha_1 \in [u_1, v_1), \cdots, \alpha_s \in [u_s, v_s), \text{ where } u_i, v_i \in \mathbb{Q}.$$

These zeros are called algebraic numbers. For infinite precision numbers, special representation mechanisms are needed. The next section presents different representations for real algebraic numbers and provides algorithms for calculations with these numbers. Finally, the base phase performs the decomposition of $\mathbb{R}^1$ into points - the zeros - and open intervals:

$$(-\infty, \alpha_1), [\alpha_1, \alpha_1], (\alpha_1, \alpha_2), \cdots, [\alpha_s, \alpha_s], (\alpha_s, \infty)$$

and selects a sample point ξ for each cell (interval) c according to:

$$\xi = \begin{cases} \alpha_1 - 1, & \text{if c} = (-\infty, \alpha_1); \\ \alpha_i, & \text{if c} = [\alpha_i, \alpha_i]; \\ (\alpha_i + \alpha_{i+1})/2, & \text{if c} = (\alpha_i, \alpha_{i+1}); \\ \alpha_s + 1, & \text{if c} = (\alpha_s, \infty). \end{cases} \tag{4.4}$$

The **extension phase** takes the decomposition of $\mathbb{R}^1$ and extends it to $\mathbb{R}^2 \cdots \mathbb{R}^n$. Sample point constructions of all the cells of the CAD of $\mathbb{R}^n$ are performed in this phase. Starting from the sample point construction of the base phase, a sample point $\xi = (\xi_1, \xi_2) \in \mathbb{R}^2$ for each cell of the stack over the cells of the base decomposition is constructed. Successive construction of sample points, repeated $n-1$ times, results finally in sample points for all cells of the CAD of $\mathbb{R}^n$. To determine whether a real polynomial system has a real solution, one only has to evaluate the signs of $\mathcal{F}$ for a sample point of each cell since $\mathcal{F}$ remains invariant on each cell by this construction.

In order to extend a sign invariant decomposition $\mathcal{D}_{i-1}$ of $\mathbb{R}^{i-1}$ to a sign invariant decompositon $\mathcal{D}_i$ of $\mathbb{R}^i$ the mathematical techniques from the base phase are applied, which means root isolation and choice of sample points. Suppose the base decompositon of $\mathbb{R}^1$ in terms of cells and sample points in one variable has been computed. Then, the polynomials of the next higher dimensional projection set, given by $PROJECTION^{n-2}(\mathcal{F}) \subset \mathbb{R}[x_1, x_2]$, are evaluated over each sample point $\xi \in c \subset \mathcal{D}^1$, where c denotes a cell. This evaluation yields a set of univariate polynomials for each sample point ξ in the variable x_2. Similar to the base phase, the roots of these polynomials are isolated and the sample points constructed for each interval of the decomposition. The received sample points represent the second component of a cell in $\mathbb{R}^2$, where the first component denotes the sample point over which the stack has been constructed. This corresponds to the geometical interpretation of raising vertical lines and stripes over the closed and open intervals of the lower dimensional decomposition and computing the intersections of these lines and stripes with the zero set of the next higher dimensional set of projection. The presented process has to be repeated until all sample points of $\mathcal{D}^3, \cdots, \mathcal{D}^n$ are completed. The CAD algorithm finally provides a list of cells and their sample points. The cylindrical algebraic decomposition can be seen as a tree structure, where the first level of nodes under the root belongs to the cells of $\mathbb{R}^1$, the second level to cells of $\mathbb{R}^2$, etc.

Now, the algorithm for cylindrical algebraic decomposition (CAD) is presented as it can be used to answer the question whether a given real polynomial system has a common solution

or not. The subsequent CAD algorithm takes the set of the polynomials from the given real polynomial system as input and returns a list of sample points of the $\mathcal{F}$-sign-invariant cylindrical algebraic decomposition of $\mathbb{R}^n$. The BASE-DECOMPOSITION algorithm which is used inside the CAD algorithm is provided here after. The REAL-ROOT-ISOLATION algorithm requires information on algebraic numbers and thus is presented in Section 4.2. To answer the decision problem, each element of the list of sample points is inserted into the real polynomial system until one element is found which satisfies the real polynomial system. If none of the sample points satisfies the real polynomial system, it does not have a common real solution.

Algorithm 4.1 CAD($\mathcal{F}$)

Input: $\mathcal{F} = f_0, \cdots, f_r \subseteq \mathbb{Q}[x_1, \cdots, x_n]$.
Output: A list of sample points of the $\mathcal{F}$-sign-invariant CAD of $\mathbb{R}^n$.
sp-ext-list := {}; sp-base-list := {}; sp-base1-list := {}; sp-base2-list := {};
if $n = 1$ **then**
  sp-base-list := BASE-DECOMPOSTION($\mathcal{F}$); {sp-base-list : samplepoint-base-list}
  return sp-base-list;
else
  $\mathcal{F}^{n-1} := PROJECTION(\mathcal{F})$;
  for $j = 2$ to $n - 1$ **do**
    $\mathcal{F}^{n-j} := PROJECTION(\mathcal{F}^{n-(j+1)})$;
  end for
  sp-base1-list := BASE-DECOMPOSITION($\mathcal{F}^1$);
  for $j = 2$ to n **do**
    n-sp-base1 := CARDINALITY(sp-base1-list);
    for $i = 1$ to n-sp-base1 **do**
      sp-base2-list := BASE-DECOMPOSITION($\mathcal{F}^j$ (sp-base1-list$_\mathrm{i}$));
      n-sp-base2 := CARDINALITY(sp-base2-list);
      for $k = 1$ to n-sp-base2 **do**
        sp-ext-list := sp-ext-list + (sp-base1-list$_\mathrm{i}$, sp-base2-list$_\mathrm{k}$);
        {sp-ext-list : samplepoint-extension-list}
      end for
    end for
    sp-base1-list := sp-ext-list;
    sp-ext-list := {};
  end for
  return sp-ext-list;
end if

EXAMPLE 4.2
This example is taken from [Jir95]. It determines the CAD in $\mathbb{R}^2$ for the set $\mathcal{F} = \{f_1, f_2\}$ with the following polynomials

$$\begin{aligned} f_1 &= x_2^2 - 2x_1x_2 + x_1^4 \\ f_2 &= (2431x_1 - 3301)x_2 - 2431x_1 + 2685. \end{aligned} \tag{4.5}$$

Algorithm 4.2 BASE-DECOMPOSITION($\mathcal{F}$)

Input: $\mathcal{F} = f_0, \cdots, f_r \subseteq \mathbb{Q}[x]$.
Output: A list of sample points for the decomposition of $\mathbb{R}^1$ according to Equation (4.4).
zero-list := {};
for $i = 0$ to r **do**
  f-zero-list := REAL-ROOT-ISOLATION(f_i);
  zero-list := zero-list + f-zero-list;
end for
zero-list := SORT(zero-list);
z := CARDINALITY(zero-list);
sp-list := sp-list + $(\alpha_0 - 1)$; {sp-list : samplepoint list}
for $i = 0$ to $z - 2$ **do**
  sp-list := sp-list + (α_i) + $((\alpha_i + \alpha_{i+1})/2)$;
end for
sp-list := sp-list + (α_{z-1}) + $(\alpha_{z-1} + 1)$;
return sp-list;

Figure 4.4 illustrates a sketch of the real zero sets of the preceeding polynomials.

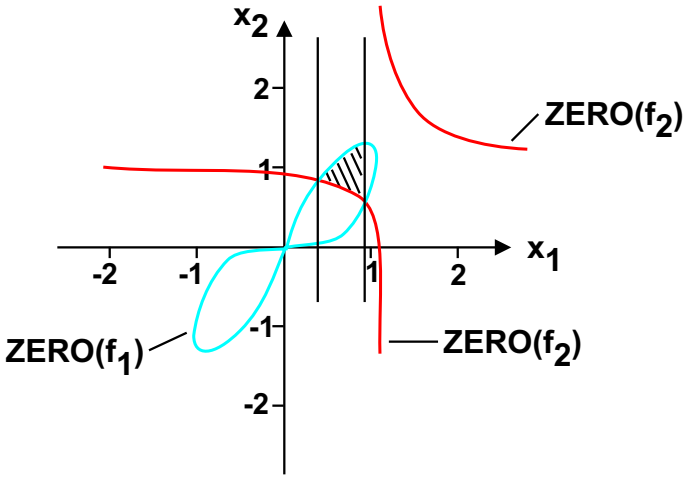

Figure 4.4: A sketch of the real zero sets of the polynomials in Corollary (4.5) (Figure taken from [Jir95]).

The CAD algorithm starts with the projection phase. First, the projection operator from Corollary (4.3) is applied. This is done only once since the example is only two-dimensional. Therefore, the set of reducta of f_1 and f_2 have to be computed first:

$$RED(f_1) = \{x_2^2 - 2x_1x_2 + x_1^4, -2x_1x_2 + x_1^4, x_1^4\} \tag{4.6}$$
$$RED(f_2) = \{(2431x_1 - 3301)x_2 - 2431x_1 + 2685, -2431x_1 + 2685\}. \tag{4.7}$$

The projection is performed in such a way that the projection operators $PROJECTION_1$ and $PROJECTION_2$ from Corollary (4.1) and from Corollary (4.2) respectively are

applied seperately to the polynomials f_1 and f_2. Hence, $PROJECTION_1(f_1)$ is equal to the union of the following sets:

$$PROJECTION_1(f_1) =$$
$$\{\{1, -2x_1, x_1^4\} \cup PSC(f_1, f_1') \cup PSC(-2x_1x_2 + x_1^4, -2x_1) \cup PSC(x_1^4, 0)\} \tag{4.8}$$

The first set is received from the head coefficients with respect to the variable x_2 of any element in the set of reducta of f_1, $RED(f_1)$, as given by Corollary (4.6). The other three sets are now computed according to Definition 2.49. A psc set consists of all i$^{\text{th}}$ principle subresultant coefficients from Corollary (2.28), where $0 < i \leq min(m, h)$ and m and h denote the degrees of the two considered polynomials. Hence, for the calculation of $PSC(f_1, f_1')$ from Corollary (4.8), f_1, f_1' are equal to $f_1 = x_2^2 - 2x_1x_2 + x_1^4$ and $f_1' = 2x_2 - 2x_1$ with $deg_{x_2}(f_1) = 2$ and $deg_{x_2}(f_1') = 1$, respectively. Therefore, the sole element of $PSC(f_1, f_1')$ is the 0^{th} principle subresultant coefficient, psc_0 since the minimum of $deg_{x_2}(f_1)$ and $deg_{x_2}(f_1')$ is 1. To calculate the i$^{\text{th}}$ principle subresultant coefficient the matrix M_i has to be constructed according to Definition 2.47. In this case, only M_0 is needed, which is equal to the Sylvestermatrix as defined by Corollary (2.25):

$$M_0 = MATRIX((f_1, x_2 \cdot f_1', f_1')) = \begin{bmatrix} 1 & -2x_1 & x_1^4 \\ 2 & -2x_1 & 0 \\ 0 & 2 & -2x_1 \end{bmatrix}.$$

Then, $PSC(f_1, f_1') = \{psc_0(f_1, f_1')\}$ with:

$$psc_0(f_1, f_1') = det(M_0^{(m+h-2i)}) = det(M_0^{(2+1-2\cdot 0)}) = det(M_0) = 4x_1^4 - 4x_1^2. \tag{4.9}$$

In the same manner $PSC(-2x_1x_2+x_1^4, -2x_1)$ from Corollary (4.8) is computed with $g_1 = -2x_1x_2+x_1^4$ and $g_1' = -2x_1$. The minimum of the degrees of the two polynomials is equal to 0 since $deg_{x_2}(g_1) = 1$ and $deg_{x_2}(g_1') = 0$. Therefore, no i$^{\text{th}}$ principle subresultant coefficient can be calculated and $PSC(-2x_1x_2 + x_1^4, -2x_1)$ results in the empty set. The same argument holds for the psc set $PSC(x_1^4, 0)$ because again both degrees of the polynomials are equal to 0. Thus no principle subresultant coefficient can be determined. Altogether, the projection of $PROJECTION_1(f_1)$ yields:

$$PROJECTION_1(f_1) = \{1, -2x_1, x_1^4, 4x_1^4 - 4x_1^2\}. \tag{4.10}$$

Now, the projection operator $PROJECTION_1$ is applied to the polynomial f_2:

$$PROJECTION_1(f_2) =$$
$$\{\{2431x_1 - 3301, -2431x_1 + 2685\} \cup PSC(f_2, f_2') \cup PSC(-2431x_1 + 2685, 0)\} \tag{4.11}$$

Again, the first set denotes the head coefficients with respect to the variable x_2 of any element in the set of reducta of f_2, $RED(f_2)$, as given by Corollary (4.7). The second psc set $PSC(-2431x_1 + 2685, 0)$ is equal to the empty set since the minimum of both degrees results in 0. $PSC(f_2, f_2')$ is computed for the polynomials $f_2 = (2431x_1 - 3301)x_2 - 2431x_1 + 2685$ and $f_2' = 2431x_1 - 3301$ with the degrees $deg_{x_2}(f_2) = 1$ and $deg_{x_2}(f_2') = 0$, respectively. The minimium of both degrees can also be calculated to be 0 and thus $PSC(f_2, f_2') = \{\}$. Finally, the projection set of $PROJECTION_1(f_2)$ is:

$$PROJECTION_1(f_2) = \{2431x_1 - 3301, -2431x_1 + 2685\}. \tag{4.12}$$

The second projection operator, $PROJECTION_2$, is now used according to Corollary (4.2). The projection contains all the psc sets of any element of the set of reducta of f_1 and f_2:

$$\begin{aligned}&PROJECTION_2(\mathcal{F}) = \\ &\{PSC(f_1, f_2) \cup PSC(f_1, -2431x_1 + 2685) \cup PSC(-2x_1x_2 + x_1^4, f_2) \cup \\ &\ PSC(-2x_1x_2 + x_1^4, -2431x_1 + 2685) \cup PSC(x_1^4, f_2) \cup PSC(x_1^4, -2431x_1 + 2685)\}\end{aligned} \tag{4.13}$$

It can be easily verified that

$$\begin{aligned}PSC(f_1, -2431x_1 + 2685) &= PSC(-2x_1x_2 + x_1^4, -2431x_1 + 2685) \\ &= PSC(x_1^4, -2431x_1 + 2685) = \{\},\end{aligned}$$

since $deg_{x_2}(-2431x_1 + 2685) = 0$ and therefore no i^{th} principle subresultant coefficient can be determined. The same is true for $PSC(x_1^4, f_2)$ since $deg_{x_2}(x_1^4) = 0$. The psc set for f_1 and f_2 with $deg_{x_2}(f_1) = 2$ and $deg_{x_2}(f_2) = 1$ consists of one single element: $PSC(f_1, f_2) = \{psc_0(f_1, f_2)\}$ since the minimum of the degree is equal to 1 and therefore the principle subresultant coefficient is only computed for $i = 0$. The Sylvester matrix M_0 is given by:

$$M_0 = MATRIX((f_1, x_2 \cdot f_2, f_2)) = \begin{bmatrix} 1 & -2x_1 & x_1^4 \\ 2431x_1 - 3301 & -2431x_1 + 2685 & 0 \\ 0 & 2431x_1 - 3301 & -2431x_1 + 2685 \end{bmatrix}$$

and $PSC(f_1, f_2)$ contains:

$$\begin{aligned}psc_0(f_1, f_2) &= det(M_0^{(m+h-2i)}) = det(M_0^{(2+1-2\cdot 0)}) = det(M_0) \\ &= (221x_1^2 - 280x_1 + 75) \cdot (26741x_1^4 - 38742x_1^3 - 8854x_1^2 - 51552x_1 + 96123).\end{aligned} \tag{4.14}$$

$PSC(-2x_1x_2 + x_1^4, f_2)$ is the last psc set to be computed from Corollary (4.13). The polynomial $g = -2x_1x_2 + x_1^4$ has degree 1 as well as the polynomial f_2 is of degree 1 with respect to the variable x_2. Thus, the psc set consists of the sole element $psc_0(-2x_1x_2 + x_1^4, f_2)$. For the given polynomials, the Sylvester matrix is constructed as follows:

$$M_0 = MATRIX((g, f_2)) = \begin{bmatrix} -2x_1 & x_1^4 \\ 2431x_1 - 3301 & -2431x_1 + 2685 \end{bmatrix}.$$

With this Sylvester matrix the 0th principle subresultant coefficient results in:

$$\begin{aligned}psc_0(g, f_2) &= det(M_0^{(m+h-2i)}) = det(M_0^{(1+1-2\cdot 0)}) = det(M_0) \\ &= -2431x_1^5 + 3301x_1^4 + 4862x_1^2 - 5370x_1.\end{aligned} \tag{4.15}$$

$PROJECTION(\mathcal{F})$ is then the sum of $PROJECTION_1(f_1)$, $PROJECTION_1(f_2)$ and $PROJECTION_2(\mathcal{F})$.

The second phase of the CAD algorithm is the base phase. First the real roots of $PROJECTION(\mathcal{F})$, which is the last and only projection set in this example, are determined. Then, the decomposition of $\mathbb{R}^1$ is performed and finally a sample point construction

for each cell of the decomposition is computed. From $PROJECTION_1(f_1)$ one receives the zeros: -1, 0 and 1. $PROJECTION_1(f_2)$ provides the two zeros: $\frac{2685}{2431}$ and $\frac{3301}{2431}$. The first term of Corollary (4.14) from $PROJECTION_2(\mathcal{F})$ has the two real roots $\frac{5}{13}$ and $\frac{15}{17}$. Corollary (4.15) from $PROJECTION_2(\mathcal{F})$ can be solved providing two real algebraic numbers as real zeros: $\alpha \approx 0.93208$ and $\beta \approx 1.59982$ as well as the real root 0. Thus, the real zeros of $PROJECTION(\mathcal{F})$ are:

$$-1, 0, \frac{5}{13}, \frac{15}{17}, \alpha \approx 0.93208, 1, \frac{2685}{2431}, \frac{3301}{2431}, \beta \approx 1.59982.$$

[Jir95] states that only six roots are needed for the construction of the CAD of $\mathbb{R}^2$. These are:

$$-1, 0, \frac{5}{13}, \frac{15}{17}, 1, \frac{3301}{2431}.$$

With these six zeros, the decomposition of $\mathbb{R}^1$ consists of 13 intervals:

$$(-\infty, -1), [-1, -1], (-1, 0), [0, 0], (0, \frac{5}{13}), [\frac{5}{13}, \frac{5}{13}], (\frac{5}{13}, \frac{15}{17}), [\frac{15}{17}, \frac{15}{17}], (\frac{15}{17}, 1), [1, 1],$$

$$(1, \frac{3301}{2431}), [\frac{3301}{2431}, \frac{3301}{2431}], (\frac{3301}{2431}, \infty).$$

The sample points for these intervals, which are selected conveniently and not according to Equation 4.4 in this example, are from left to right as follows:

$$-2, -1, -\frac{1}{2}, 0, \frac{1}{4}, \frac{5}{13}, \frac{1}{2}, \frac{15}{17}, \frac{9}{10}, 1, \frac{5}{4}, \frac{3301}{2431}, 2.$$

In the last phase of the CAD algorithm, the extension phase, the decomposition of $\mathbb{R}^1$ is lifted to a decomposition of $\mathbb{R}^2$ by constructing the stack of each cell of the base decomposition. The whole CAD consists of 63 cells. In this example, only the calculation of the stack over cell 7, $\frac{5}{13}, \frac{15}{17}$), with the sample point $x_1 = \frac{1}{2}$ is illustrated. Therefore, the sample point is inserted into the two polynomials f_1 and f_2 of $\mathbb{R}^2$, yielding two univariate polynomials in the variable x_2:

$$f_1(\frac{1}{2}, x_2) = x_2^2 - x_2 + \frac{1}{16}$$
$$f_2(\frac{1}{2}, x_2) = -\frac{4171}{2}x_2 + \frac{2939}{2}.$$

The real root isolation process results in three zeros:

$$\frac{1}{2} \pm \frac{\sqrt{3}}{4}, \textit{ and } \frac{2939}{4171}.$$

For the decomposition of $\mathbb{R}^2$ over the cell 7 one gets 7 intervals:

$$(-\infty, \frac{1}{2} - \frac{\sqrt{3}}{4}), [\frac{1}{2} - \frac{\sqrt{3}}{4}, \frac{1}{2} - \frac{\sqrt{3}}{4}], (\frac{1}{2} - \frac{\sqrt{3}}{4}, \frac{2939}{4171}), [\frac{2939}{4171}, \frac{2939}{4171}],$$

$$(\frac{2939}{4171}, \frac{1}{2} + \frac{\sqrt{3}}{4}), [\frac{1}{2} + \frac{\sqrt{3}}{4}, \frac{1}{2} + \frac{\sqrt{3}}{4}], (\frac{1}{2} + \frac{\sqrt{3}}{4}, \infty).$$

The following sample points chosen for the 7 intervals, which are again selected conveniently and not according to Equation 4.4 in this example, are:

$$0, \frac{1}{2} - \frac{\sqrt{3}}{4}, \frac{1}{2}, \frac{2939}{4171}, \frac{3}{4}, \frac{1}{2} + \frac{\sqrt{3}}{4}, 2.$$

The following table presents the sample points for the 7 cells received from the stack over cell 7 of the base decomposition and the corresponding signs of the polynomials f_1 and f_2 within these cells.

Sample point (x_1, x_2)	$sign(f_1)$	$sign(f_2)$
$(\frac{1}{2}, 0)$	+	+
$(\frac{1}{2}, \frac{1}{2} - \frac{\sqrt{3}}{4})$	0	+
$(\frac{1}{2}, \frac{1}{2})$	-	+
$(\frac{1}{2}, \frac{2939}{4171})$	-	0
$(\frac{1}{2}, \frac{3}{4})$	-	-
$(\frac{1}{2}, \frac{1}{2} + \frac{\sqrt{3}}{4})$	0	-
$(\frac{1}{2}, 2)$	+	-

Table 4.1: Sample points and signs of f_1 and f_2 over cell 7 (Table taken from [Jir95]).

Suppose the real polynomial system $f_1 < 0$ and $f_2 < 0$. This real polynomial system has a common real solution if one can determine a cell in the CAD, where the sign of f_1 is negative and also the sign of f_2 is negative. This is given in this example for the cell in row five of Table 4.1. This cell corresponds to the striped region in Figure 4.4.

4.2 Calculations with Real Algebraic Numbers

This section deals with some representations of real algebraic numbers and integers as well as with how to calculate with these numbers. Real algebraic numbers and integers solve the problem of how to represent an infinite precision number in limited time space and computer memory. Infinite precision numbers occur in the isolation procedure for real roots of univariate polynomials which is needed in the base and in the extension phase of the CAD algorithm.

First, the terms algebraic number and algebraic integer are defined. Then, three different representations of real algebraic numbers are considered. Finally, the real-root-isolation problem is solved and calculations with real algebraic numbers are discussed.

DEFINITION 4.11
*The zero of a polynomial $f \in \mathbb{Z}[x]$ is called an **algebraic number** α. f is said to be a **minimal polynomial** of α if the degree of f is minimal. In order to make f unique, f has to be primitive and its leading coefficient has to be a distinguished element of $\mathbb{Z}$. The **degree** of α is defined as the degree of its minimal polynomial. A nonalgebraic number in $\mathbb{C}$ is called a **transcendental number**. An algebraic number α is **integral** or an **algebraic integer** if α is the root of a monic integer polynomial. **Real algebraic numbers** define the subset of the set of algebraic numbers for which $\alpha \in \mathbb{R}$ is true. The set of all real algebraic numbers is denoted by $\mathbb{A}$.*

In general, algebraic numbers are elements in $\mathbb{C}$. For example, $\frac{1}{2}(1+\sqrt{5})$ is an algebraic integer since it is a zero of the monic polynomial $x^2 - x - 1$. Ordinary integers $n \in \mathbb{Z}$ are algebraic integers, as they are roots of $x - n = 0$. Also, all rational numbers $\frac{p}{q} \in \mathbb{Q}\,(q \neq 0)$ constitute to real algebraic numbers since they are zeros of the polynomials $qx - p = 0$. The zeros of the polynomials $qx^n - p = 0$ are n-dimensional square roots of rational numbers in $\mathbb{R}$ which also represent real algebraic numbers. π (circumference of the circle with unit diameter) and e (base of the natural logarithm) pose two examples for transcendental numbers. The minimal polynomial for $\alpha = 0$ is the zero polynomial which degree is equal to $-\infty$.

The following two lemmas are repeated here without any proof. A proof is given in [Mis93].

LEMMA 4.3 *Every (real) algebraic number can be expressed as a (real) algebraic integer divided by an integer.*

LEMMA 4.4 *If α, β are real algebraic numbers (real algebraic integers), then so are $-\alpha$, α^{-1} $(\alpha \neq 0)$, $\alpha + \beta$ and $\alpha \cdot \beta$.*

The coding of real algebraic numbers can be done in three different ways. One of them is based on Thom's Lemma which is explained in the following. The proof is presented in [Yap00].

LEMMA 4.5 (Thom's Lemma) *Let $f \in \mathbb{R}[x]$ be a real univariate polynomial of degree n and let $f', \cdots, f^{(i)}$ be its derivatives. Given a sign condition $\varepsilon = (\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_n) \in \{-1, 0, +1\}^{n+1}$, $A(\varepsilon)$ is defined as follows:*

$$A(\varepsilon) = \{x \in \mathbb{R} \mid sign(f^{(i)}(x)) = \varepsilon_i, for\ i = 0, \cdots, n\}, \tag{4.16}$$

where $f^{(i)}(x)$ denotes the polynomial function of $f^{(i)}$. Then, $A(\varepsilon)$ is either empty or connected.

Thom's Lemma implies that every non-empty set $A(\varepsilon)$ must be connected, that means it consists of a single interval. A real root α of a polynomial f of degree n with real coefficients may be distinguished from the other real roots of f by the signs of the derivatives $f^{(i)}$ of f at α, for $i = 0, \cdots, n$. The next Corollary characterizes a real algebraic number using Thom's Lemma, which is proved in [Yap00] or in [CR88].

COROLLARY 4.6 *Let f be a polynomial of degree n with integer coefficients. Let α and α' be two distinct real roots of f. Suppose $\varepsilon = (\varepsilon_0, \varepsilon_1, \cdots, \varepsilon_n)$ and $\varepsilon' = (\varepsilon'_0, \varepsilon'_1, \cdots, \varepsilon'_n)$ to be the sign conditions of $f^{(i)}(\alpha)$ and $f^{(i)}(\alpha')$, respectively. Then:*

(i) ε and ε' are distinct.

(ii) Let i be the largest index such that $\varepsilon_i \neq \varepsilon'_i$. Then $\varepsilon_{i+1} = \varepsilon'_{i+1} \neq 0$, for $0 < i < n$. Furthermore, $\alpha < \alpha'$ iff one of the following conditions holds:

(a) $\varepsilon_{i+1} = +1 \quad and \quad \varepsilon_i < \varepsilon'_i$;

(b) $\varepsilon_{i+1} = -1 \quad and \quad \varepsilon_i > \varepsilon'_i$.

Corollary 4.6 is a direct consequence of Thom's Lemma. It defines a real algebraic number and implies how to decide whether $\alpha < \alpha'$ or $\alpha > \alpha'$ for to given distinct real roots α and α'. Basically, a real algebraic number ξ can be represented by a polynomial $f \in \mathbb{Z}[x]$ for which α is a zero and by additional information identifying this particular zero. Three different representations of real algebraic numbers are given now.

COROLLARY 4.7 *Let f be a polynomial of degree n with integer coefficients and assume that the distinct real roots of f are enumerated as follows:*

$$\alpha_1 < \alpha_2 < \cdots < \alpha_{i-1} < \alpha_i = \alpha < \alpha_{i+1} < \cdots < \alpha_l,$$

where $l \leq n$.

1. ***Order Representation**: A real algebraic number is represented as a pair consisting of its polynomial f and its index i which determines the real root within the sequence of enumerated real roots of f:*

   $$\alpha_{order} = \langle f, i \rangle.$$

   This representation requires $O(n \, lg \, \|f\|_1 + log \, n)$ bits.

2. ***Interval Representation**: A real algebraic number is represented as a triple consisting of its polynomial f and the two end points of an isolation interval (l, r) $(l, r \in \mathbb{Q}, l < r)$ which contains only α as a zero:*

   $$\alpha_{interval} = \langle f, l, r \rangle.$$

   The interval representation needs only $O(n \, lg \, \|f\|_1 + n \, lg \, n)$ bits.

3. ***Sign Representation**: A real algebraic number is represented as a pair consisting of its polynomial f and the sign condition ε indicating the signs of the derivatives $f^{(i)}$ of f at α:*

   $$\alpha_{sign} = \langle f, \varepsilon \rangle.$$

   The sign representation requires only $O(n \, lg \, \|f\|_1 + n)$ bits.

EXAMPLE 4.3
This example illustrates the three different representations for the real algebraic number $\alpha = \sqrt{2} + \sqrt{3}$:

$$\begin{aligned}
\alpha_{order} &= \sqrt{2} + \sqrt{3} = \langle x^4 - 10x^2 + 1, 4 \rangle, \\
\alpha_{interval} &= \sqrt{2} + \sqrt{3} = \langle x^4 - 10x^2 + 1, 3, 7/2 \rangle, \\
\alpha_{sign} &= \sqrt{2} + \sqrt{3} = \langle x^4 - 10x^2 + 1, (+1, +1, +1) \rangle.
\end{aligned}$$

The Algorithm 4.3 solves the real root isolation problem for a given polynomial and returns an ordered list of isolating intervals. It isolates all distinct zeros of an integral polynomial $f(x)$ by computing an isolating interval for each zero. The main mathematical tool for this algorithm is taken from Corollary 2.13. It implies a method for calculating the number of distinct real roots within a chosen interval by applying the Standard Sturm Sequence from

Algorithm 4.3 REAL-ROOT-ISOLATION(f)

Input: $f \in \mathbb{Z}[x]$;
Output: An isolating interval (a,b) $(a, b \in \mathbb{Q})$, containing a real root of f).
$h_1, \cdots, h_p := STURM(f, f')$;
$a := -NORM_1(f)$; $b := NORM_1(f)$; $s := [2 \cdot n^{(n/2)+2}(NORM_\infty(f) + 1)]^{-1}$;
rootinterval-list := { };
roots := $Var((h_1, h_2, \cdots, h_p)(a)) - Var((h_1, h_2, \cdots, h_p)(b))$;
if $Var((h_1, h_2, \cdots, h_p)(a)) = Var((h_1, h_2, \cdots, h_p)(b))$ **then**
  return "No root in (a, b)";
else
  for $i = 0$ to roots **do**
    while $Var((h_1, h_2, \cdots, h_p)(a)) - Var((h_1, h_2, \cdots, h_p)(b)) > 1$ **do**
      $c := (a + b)/2$;
      if $Var((h_1, h_2, \cdots, h_p)(a)) > Var((h_1, h_2, \cdots, h_p)(c))$ **then**
        $b := c$;
      else
        $a := c$;
      end if
    end while
    if $f(b) = 0$ **then**
      rootinterval-list := rootinterval-list + { $[b, b]$ };
      $a := b + s$;
    else
      rootinterval-list := rootinterval-list + { (a, b) };
      $a := b$;
    end if
    $b := NORM_1(f)$;
  end for
end if
return rootinterval-list ;

Definition 2.52. A starting interval is shortened using biselection until it contains only one distinct zero of the polynomial. The second interval bound has to be checked whether it is itself a real root of $f(x)$, whereas the first interval bound does not have to be checked for zero as illustrated in Example 2.14. In case the second interval bound is a root itself, the closed interval $[b, b]$ is appended to the interval list of the roots and the lower interval bound is set to $a := b + s$, where s indicates the minimal separation bound according to Corollary 2.4. If $s \notin \mathbb{Q}$, s has to be set to the next smaller rational number. (This is not implemented in the algorithm yet.) Thus, it is ensured that the lower interval bound is not a root itself. If the second interval bound is not a zero itself, the lower interval bound is adjusted such that it equals the higher bound from the previous interval. In both cases, the higher interval bound is reset to the starting value. The starting interval is defined according to Theorem 2.3. The algorithm finds the isolating intervals for the zeros of the polynomial from the left to the right of the starting interval and returns all the intervals in a list.

For the interval representation, a normalization has to be performed to receive that if $\alpha \neq 0$ then $0 \notin (l, r)$, so that l and r have the same sign. The normalization can be achieved by the Algorithm 4.4. Remember that the bound p is defined according to Theorem 2.3. p denotes the smallest positive bound over 0 and $-p$ indicates the greatest negative bound below 0. Which case is true is checked using Corollary 2.13. The algorithm has a time complexity of $O(n^2)$ with $deg(f) = n$.

Algorithm 4.4 NORMALIZATION(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: A representation of $\alpha = \langle f, l', r'\rangle$ such that $0 \notin (l', r')$
$S(x) := STURM(f(x), f'(x))$;
$p := 1/(1 + \|f\|_\infty)$;
if $Var(S(l)) > Var(S(-p))$ **then**
  $\alpha = \langle f, l, -p\rangle$;
else if $Var(S(-p)) > Var(S(p))$ **then**
  $\alpha = 0$;
else
  $\alpha = \langle f, p, r\rangle$;
end if
return α;

The REFINE algorithm simply bisects the input interval of the real algebraic number and returns the halves which still includes the zero. Here also, the time complexity can be shown to be in the order of $O(n^2)$.

Algorithm 4.5 REFINE(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: A finer representation of $\alpha = \langle f, l', r'\rangle$ such that $2(r' - l') \leq (r - l)$
$h_1, \cdots, h_p := STURM(f, f')$;
$m := (l + r)/2$;
if $Var((h_1, h_2, \cdots, h_p)(l)) > Var((h_1, h_2, \cdots, h_p)(m))$ **then**
  $\alpha = \langle f, l, m\rangle$;
else
  $\alpha = \langle f, m, r\rangle$;
end if
return α;

The SIGN algorithm evaluates the sign of a univariate polynomial at a real algebraic number. The algorithm is a straight application of the Sturm-Sylvester Theorem, Theorem 2.12. It is integrated in the Interval-to-Sign conversion algorithm which will be presented later in this section. Furthermore, the sign evaluation algorithm can be used to compare an algebraic number α with a rational number $\frac{p}{q}$ by computing the sign of $qx - p$ at α. Another application could be the calculation of the multiplicity of a root α of a polynomial f by evaluating the sign of the polynomials $f', f^{(2)}, f^{(3)}, \cdots$ at α. The time complexity is given by $O(n^2)$, [Mis93].

Algorithm 4.6 SIGN(α, g)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$ and a univariate rational polynomial $g(x) \in \mathbb{Q}[x]$.
Output: $sign(g(\alpha))$ = sign of g at α
$h_1, \cdots, h_p := STURM(f, f'g)$;
return $Var((h_1, h_2, \cdots, h_p)(l)) - Var((h_1, h_2, \cdots, h_p)(r))$;

The interval representation of a real algebraic number is used in many more applications compared to the order or sign represetation. For the given problem of the cylindrical algebraic decomposition of $\mathcal{R}^n$, the interval representation seems to be the obvious one to apply to. In [CR88], the important topic of real roots of a polynomial with real algebraic number coefficients is addressed using sign representation. Therefore, the Interval-to-Order conversion algorithm is presented, followed by the Interval-to-Sign conversion algorithm. The time complexities for the Interval-to-Order and Interval-to-Sign algorithm are stated by [Mis93] with $O(n^2)$ and $O(n^3)$, respectively.

Algorithm 4.7 INTERVAL-TO-ORDER(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: Its order representation $\alpha = \langle f, i\rangle$.
$h_1, \cdots, h_p := STURM(f, f')$;
return $\langle f, Var((h_1, h_2, \cdots, h_p)(-1 - \|f\|_\infty)) - Var((h_1, h_2, \cdots, h_p)(r))\rangle$;

Algorithm 4.8 INTERVAL-TO-SIGN(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: Its sign representation $\alpha = \langle f, \varepsilon\rangle$.
$FOURIER(f) := \langle f, f', f^{(2)}, \cdots, f^{(deg(f))}\rangle$;
$\varepsilon := (SIGN(\alpha, f), SIGN(\alpha, f'), \cdots, SIGN(\alpha, f^{(deg(f))}))$;
return $\langle f, \varepsilon\rangle$;

In order to state the correctness of an algorithm for addition and multiplication as well as an algorithm for the additive and multiplicative inverse, the following properties of resultants are needed. The theory of resultants provides several basic properties of algebraic numbers and is taken from [Yap00]. In the following, $f(x)$ and $g(x)$ are the polynomial functions corresponding to the polynomials f and g, respectively.

LEMMA 4.8 *Let $f, g \in K[x]$ with $deg(f) = m$, $deg(g) = n$ and let $\alpha, \beta \in K$. Then the following conditions hold:*

(i) $res(\alpha, g) = \alpha^n$. *By definition,* $res(\alpha, \beta) = 1$.

(ii) $res((x - \alpha) \cdot f, g) = g(\alpha) res(f, g)$.

(iii) $res(f, g) = (-1)^{mn} res(g, f)$.

(iv) $res(\alpha \cdot f, g) = \alpha^n res(f, g)$.

THEOREM 4.9 *Let $f, g \in K[x]$, $a = hoef(f)$, $b = hcoef(g)$, $deg(f) = m$, $deg(g) = n$ with roots*

$$\alpha_1, \cdots, \alpha_m, \beta_1, \cdots, \beta_n \in K.$$

Then $res(f, g)$ is equal to each of the following expressions:

(i) $a^n \prod_{i=1}^{m} g(\alpha_i)$

(ii) $(-1)^{nm} b^m \prod_{j=1}^{n} f(\beta_j)$

(iii) $a^n b^m \prod_{i=1}^{m} \prod_{j=1}^{n} (\alpha_i - \beta_j)$.

Proof:
Assume $f = a \prod_{i=1}^{m}(x - \alpha_i)$ and $g = b \prod_{j=1}^{n}(x - \beta_j)$, then condition (i) follows from Lemma 4.8 using the items in the sequence $(iv), (ii), \cdots, (ii)$.

$$res(f, g) = a^n res\left[\prod_{i=1}^{m}(x - \alpha_i), g\right] = a^n g(\alpha_1) res\left[\prod_{i=2}^{m}(x - \alpha_i), g\right] = a^n g(\alpha_1) \cdots g(\alpha_m).$$

Condition (ii) is deduced similarily. One can get (iii) from (i) since $g(\alpha_i) = b \prod_{j=1}^{n}(\alpha_i - \beta_j)$.

□

If f, g are multivariate polynomials in $K[x_1, \cdots, x_n]$, their resultant can be calculated by considering them as univariate polynomials in any variable $y = x_i$. This is then indicated by a subscript $res_y(f, g)$.

LEMMA 4.10 *Let $f, g \in K[x]$, $\alpha, \beta \in K$ and $deg(f) = m$, $deg(g) = n$. If α is a root of f, β a root of g which are not equal to 0, then the next conditions are true:*

(i) *$1/\alpha$ is the root of $x^m f(1/x)$ provided $\alpha \neq 0$.*

(ii) *$\beta \pm \alpha$ is a root of $h(x) = res_y(f(y), g(x \mp y))$.*

(iii) *$\alpha\beta$ is a root of $h(x) = res_y(f(y), y^n g(\frac{x}{y}))$.*

Proof:
Assume again that $f = a \prod_{i=1}^{m}(z - \alpha_i)$ and $g = b \prod_{j=1}^{n}(z - \beta_j)$.

(i) This is immediate.

(ii) It follows from Theorem 4.9 (i) and $g(x \mp \alpha_i) = b \prod_{j=1}^{n}(x \mp \alpha_i - \beta_j)$ that:

$$res_y\left[f(y), g(x \mp y)\right] = a^n \prod_{i=1}^{m} g(x \mp \alpha_i) = a^n b^m \prod_{i=1}^{m} \prod_{j=1}^{n} (x \mp \alpha_i - \beta_j).$$

(iii) This is deduced from Theorem 4.9 (i) and $g(\frac{x}{\alpha_i}) = b\prod_{j=1}^{n}(\frac{x}{\alpha_i} - \beta_j)$ followed by some transformations.

$$res_y\left[f(y), y^n g(\frac{x}{y})\right] = a^n \prod_{i=1}^{m}\left[\alpha_i^n g(\frac{x}{\alpha_i})\right] = a^n \prod_{i=1}^{m}\left[b\alpha_i^n \prod_{j=1}^{n}(\frac{x}{\alpha_i} - \beta_j)\right]$$
$$= a^n b^m \prod_{i=1}^{m}\prod_{j=1}^{n}(x - \alpha_i\beta_j).$$

□

Now, the algorithms for some arithmetic operations are presented.

Algorithm 4.9 ADDITIVE-INVERSE(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: $-\alpha$ and its interval representation.
return $\langle f(-x), -r, -l\rangle$;

The correctness of the algorithm for the additive-inverse of a real algebraic number can be deduced from the fact that if α is a root of $f(x)$, then $-\alpha$ is a root of $f(-x)$. The algorithm has a linear time complexity.

Algorithm 4.10 MULTIPLICATIVE-INVERSE(α)

Input: A real algebraic number $\alpha = \langle f, l, r\rangle \in \mathbb{A}$.
Output: $1/\alpha$ and its interval representation.
return $\langle x^{deg(f)} f(\frac{1}{x}), \frac{1}{r}, \frac{1}{l}\rangle$;

This preceeding algorithm follows directly from Lemma 4.10 (i). Again, this algorithm has a linear time complexity. The next algorithm is concerned with the addition of two algebraic numbers. It is implemented according to the second main property of Lemma 4.10 whose correctness has been proven above. An isolating interval is obtained by applying the refinement Algorithm 4.5 repeatedly. The size complexities of the polynomial f_3 with $deg(f_3) \leq nm$ and $deg(f_1) = n$, $deg(f_2) = m$ is given by $\|f_3\|_1 \leq 2^{O(nm)}\|f_1\|_1^m\|f_2\|_1^n$. It can be proven that the time complexity of the algorithm is in the order of $O(n^3m^4lg\|f_1\|_1 + n^4m^3lg\|f_2\|_1)$.

The algorithm for the multiplication of two real algebraic numbers can be obtained immediately from Lemma 4.10 condition (iii). The proof of its correctness has been shown above. Again, the refinement process using Algorithm 4.5 provides an isolated interval. Again, [Mis93] states a size complexity of the polynomial f_3 with $deg(f_3) \leq nm$ of $\|f_3\|_1 \leq nm\|f_1\|_1^m\|f_2\|_1^n$, where $deg(f_1) = n$ and $deg(f_2) = m$. Hence, the time complexity of the multiplication algorithm is in the same order as the addtion algorithm: $O(n^3m^4lg\|f_1\|_1 + n^4m^3lg\|f_2\|_1)$.

Before the computation of the zeros of a polynomial with real algebraic coefficients is explained, the term of number fields has to be introduced. It should be mentioned that the roots of a polynomial with algebraic number coefficients are algebraic numbers again.

Algorithm 4.11 ADDITION(α_1, α_2)

Input: Two real algebraic numbers $\alpha_1 = \langle f_1, l_1, r_1\rangle$ and $\alpha_2 = \langle f_2, l_2, r_2\rangle \in \mathbb{A}$.
Output: $\alpha_3 = \alpha_1 + \alpha_2 = \langle f_3, l_3, r_3\rangle$ and its interval representation.
$f_3 := RESULTANT_y(f_1(y), f_2(x-y))$;
$h_1, \cdots, h_p := STURM(f_3, f_3')$;
$l_3 := l_1 + l_2$;
$r_3 := r_1 + r_2$;
while $Var((h_1, h_2, \cdots, h_p)(l_3)) - Var((h_1, h_2, \cdots, h_p)(r_3)) > 1$ **do**
  $\alpha_1 := REFINE(\langle f_1, l_1, r_1\rangle)$;
  $\alpha_2 := REFINE(\langle f_2, l_2, r_2\rangle)$;
  $l_3 := l_1 + l_2$;
  $r_3 := r_1 + r_2$;
end while
return $\langle f_3, l_3, r_3\rangle$;

Algorithm 4.12 MULTIPLICATION(α_1, α_2)

Input: Two real algebraic numbers $\alpha_1 = \langle f_1, l_1, r_1\rangle$ and $\alpha_2 = \langle f_2, l_2, r_2\rangle \in \mathbb{A}$.
Output: $\alpha_3 = \alpha_1 \cdot \alpha_2 = \langle f_3, l_3, r_3\rangle$ and its interval representation.
$f_3 := RESULTANT_y(f_1(y), y^{deg(f_2)} f_2(\frac{x}{y}))$;
$h_1, \cdots, h_p := STURM(f_3, f_3')$;
$l_3 := min(l_1l_2, l_1r_2, r_1l_2, r_1r_2)$;
$r_3 := max(l_1l_2, l_1r_2, r_1l_2, r_1r_2)$;
while $Var((h_1, h_2, \cdots, h_p)(l_3)) - Var((h_1, h_2, \cdots, h_p)(r_3)) > 1$ **do**
  $\alpha_1 := REFINE(\langle f_1, l_1, r_1\rangle)$;
  $\alpha_2 := REFINE(\langle f_2, l_2, r_2\rangle)$;
  $l_3 := min(l_1l_2, l_1r_2, r_1l_2, r_1r_2)$;
  $r_3 := max(l_1l_2, l_1r_2, r_1l_2, r_1r_2)$;
end while
return $\langle f_3, l_3, r_3\rangle$;

THEOREM 4.11 *Let α be an algebraic number over the field $\mathbb{Q}$ and let f be its defining polynomial of degree $m > 0$. Then the set of all algebraic numbers represented by $\beta = \sum_{i=0}^{m-1} b_i\alpha^i = g(\alpha)$, where $b_i \in \mathbb{Q}$, forms a field. $\mathbb{Q}(\alpha)$ and subfields of the form $\mathbb{Q}(\alpha)$ are called **number fields** or **simple algebraic extensions** of $\mathbb{Q}$.*

$\mathbb{Q}(\alpha)$ is called **separable**, if the defining polynomial for α is square free. $\mathbb{Q}(\alpha)(\beta) = \mathbb{Q}(\alpha, \beta)$ is then a double extension of $\mathbb{Q}$ or a double number field, which is a field again.

THEOREM 4.12 *Every separable multiple algebraic extension can be reduced to a simple algebraic extension.*

To understand the basic arithmetic in number fields $\mathbb{Q}[\alpha]$, the minimal defining polynomial $f(x)$ of α is considered. Suppose α is of degree d. Then, it follows from $f(\alpha) = 0$ that α^d can be expressed as a polynomial of degree at most $d-1$ in α and coefficients in $\mathbb{Q}$. Every element $\beta = \sum_{i=0}^{m-1} b_i\alpha^i \in \mathbb{Q}[\alpha]$ is uniquely determined by its coefficient vector $(b_0, b_1, \cdots, b_{d-1})$. Thus, addition and subtraction in $\mathbb{Q}[\alpha]$ can be computed componentwise

in these coefficients. Multiplication is performed as in polynomial multiplication, where the received polynomial is reduced to the degree at most $d-1$.

In the extension phase of the CAD algorithm, the decomposition of $\mathbb{R}^1$ is lifted to $\mathbb{R}^2 \cdots \mathbb{R}^n$. Therefore, a sign invariant decomposition $\mathcal{D}_{i-1}$ of $\mathbb{R}^{i-1}$ is extended to a sign invariant decomposition of $\mathcal{D}_i$ of $\mathbb{R}^i$ using the technique of the base phase. A sample point $\xi = (\xi_1, \cdots, \xi_{i-1}) \in \mathbb{A}^{i-1}$ is inserted into the polynomials of the next higher dimensional projection set $\mathcal{F}^j = f_{j,1}, \cdots, f_{j,p} \subseteq \mathbb{Q}[x_1, \cdots, x_i]$, where $1 \leq j \leq n-1$. Assume the following polynomial

$$f = c_n x_i^n + \cdots + c_0,$$

with $c_n, \cdots, c_0 \in \mathbb{Q}[x_1, \cdots, x_{i-1}]$. Applying the arithmetic for real algebraic numbers, the insertion of the above sample point yields the polynomial $f(\xi, x_i) \in \mathbb{A}[x_i]$ with real algebraic number coefficients $c_i(\xi) = \langle f_i, l_i, r_i \rangle \in \mathbb{A}$. Each coefficient c_i represents a real algebraic number α_i such that $f \in \mathbb{Q}(\alpha_n, \cdots, \alpha_0)[x_i]$. $\mathbb{Q}(\alpha_n, \cdots, \alpha_0)$ is a multiple number field which can be reduced to a simple number field $\mathbb{Q}(\alpha)$ according to Theorem 4.12. A repeated reduction for the coefficients of f results in a univariate polynomial $\hat{f} \in \mathbb{Q}(\alpha)[x_i]$ whose coefficients are polynomials in α. To find the zeros of $\hat{f}$, one computes a polynomial $\hat{g} \in \mathbb{Z}[x_i]$, such that $ZERO(\hat{f}) \subseteq ZERO(\hat{g})$. The isolating intervals of the zeros of $\hat{f}$ have to be found in order to identify the corresponding zeros in the set of zeros of $\hat{g}$. A refinement might be needed for the isolating intervals of the zeros of $\hat{g}$ since the isolating intervals have to contain the same real algebraic number for $\hat{f}$ and for $\hat{g}$.

In [Loo82] an algorithm is presented that computes such a polynomial $\hat{g} \in \mathbb{Z}[x_i]$ from a polynomial $f \in \mathbb{Q}(\alpha_n, \cdots, \alpha_0)[x_i]$. The algorithm NORMAL uses the algorithm SIMPLE to perform the reduction of the multiple number field to a simple number field. The algorithm SIMPLE takes two algebraic numbers α and β and their defining polynomials as input and computes a real algebraic number γ and its defining polynomial such that $\alpha, \beta \in \mathbb{Q}(\gamma)$. The algorithm SIMPLE is given first, followed by the algorithm NORMAL.

The algorithm SIMPLE constructs the primitive element γ by applying Lemma 4.10, condition (ii). Thus, the polynomial $r(x,t)$ has the zeros $\gamma_{ij} = \alpha_i + t\beta_j$. Then, the smallest positive integer t_1 is computed, such that the polynomial $r(x, t_1)$ is square free. The condition for a square free polynomial is that the polynomial and its derivate does not have a greatest common divisor: $deg(gcd(r(x,t_1), r'(x,t_1))) \neq 0$. This implies that all γ_{ij} are different for $1 \leq i \leq m$, $1 \leq j \leq n$. Since there would be only finitely many pairs of γ_{ij} to compare but infinitely many positive integers, a t_1 can always be found. (Therefore, the WHILE-loop is not an endless loop.) In the next step, an isolating interval for the defining polynomial of γ is computed using interval arithmetic. The polynomial $h(x)$ is defined as $gcd(f_1(\gamma - t_1 x), f_2(x))$ because β_j is then a root of $h(x)$. By construction, there is only one such β. Thus, $h(x) = x - \beta$ and $g_2(x)$ equal to the negative trailing coefficient of $h(x)$. $g_1(x)$ follows then from $g_2(x)$ with $g_1(x) := x - t_1 g_1(x)$. The computing time of this algorithm is polynomial. The limiting factor of the algorithm SIMPLE is the fact that the degree of γ is growing rapidely ($deg(\gamma) = n^2$ mostly), if applied multiple times.

The basic idea of the algorithm NORMAL is deduced from Theorem 4.13.

THEOREM 4.13 *Let* $f(x) = \sum_{i=0}^{m} f_i x^i = a_m \prod_{i=1}^{m}(x - \alpha_i)$ *be a primitive square free polynomial over integral domain* R. *Let* $g(x,y) = \sum_{j=0}^{n} g_j(x) y^j$ *be a bivariate*

Algorithm 4.13 SIMPLE(α, β)

Input: Two real algebraic numbers $\alpha = \langle f_1, l_1, r_1 \rangle$ and $\beta = \langle f_2, l_2, r_2 \rangle \in \mathbb{A}$.
Output: $\gamma = \langle f_3, l_3, r_3 \rangle$ and $g_1(x), g_2(x)$ such that $\alpha = g_1(\gamma)$ and $\beta = g_2(\gamma)$.
$r(x,t) := RESULTANT_y(f_2(y), f_1(x - ty))$;
$t_1 := 1$;
while $DEG(GCD(r(x,t_1), r'(x,t_1))) \neq 0$ **do**
  $t_1 := t_1 + 1$;
end while
$f_3(x) := r(x,t_1)$;
$h_1, \cdots, h_p := STURM(f_3, f_3')$;
$l_3 := l_1 + t_1 \cdot l_2$;
$r_3 := r_1 + t_1 \cdot r_2$;
while $Var((h_1, h_2, \cdots, h_p)(l_3)) - Var((h_1, h_2, \cdots, h_p)(r_3)) > 1$ **do**
  $\alpha_1 := REFINE(\langle f_1, l_1, r_1 \rangle)$;
  $\alpha_2 := REFINE(\langle f_2, l_2, r_2 \rangle)$;
  $l_3 := l_1 + t_1 \cdot l_2$;
  $r_3 := r_1 + t_1 \cdot r_2$;
end while
$k(x) := GCD(f_1(\gamma - t_1 x), f_2(x))$;
$g_2(\gamma) := x - k(x)$;
$SUBST(x, \gamma, g_2(\gamma))$; {substitute x for γ in $g_2(\gamma)$}
$g_1(x) := x - t_1 g_1(x)$;
return $\{\langle f_3, l_3, r_3 \rangle, g_1(x), g_2(x)\}$;

polynomial over R such that $deg(gcd(f(x), g_n(x))) = 0$. Let $k = deg_y(g(x,y))$ and $r(x) = res_x(f(x), g(x,y))$. Then $r(x)$ contains the zeros of $g(x)$ among its own roots.

Proof:
It follows from Theorem 4.9 condition (i) and $g(x,y) = \sum_{j=0}^{n} g_j(x) y^j = g_n(x) \prod_{j=1}^{n} (y - \beta_j)$, which is equal to $g(\alpha_i, y) = g_n(\alpha_i) \prod_{j=1}^{n} (y - \beta_{ij})$ inserting α_i for x, that:

$$r(x) = res_x(f(x), g(x,y)) = a_m^k \prod_{i=1}^{m} g(\alpha_i, y) = a_m^k \prod_{i=1}^{m} g_n(\alpha_i) \prod_{i=1}^{n} (y - \beta_{ij}).$$

□

The isolating intervals of the zeros of $\hat{f}$ could be computed using a modified version of the REAL-ROOT-ISOLATION algorithm, Algorithm 4.3, for the number field $\mathbb{Q}[\alpha]$. Applying the REFINE algorithm, Algorithm 4.5, the isolating intervals can be adjusted such that they contain the same real algebraic number for $\hat{f}$ and for $\hat{g}$.

In [Sei01], a different representation of real algebraic numbers is presented and implemented in order to solve the decision problem for real polynomial systems using the cylindrical algebraic decomposition method of Collins. This representation has been invented to avoid arithmetics in $\mathbb{Q}[\alpha]$, which seem to pose severe limitations. The interval representation as defined in Corollary 4.7 is extended such that the defining polynomial is exchanged for a tuple. This tuple consists of a now multivariate polynomial in $\mathbb{Q}$ in the

Algorithm 4.14 NORMAL($f(\alpha_n, \cdots, \alpha_0, x_i)$)

Input: A polynomial $f(\alpha_n, \cdots, \alpha_0, x_i)$, where $\alpha_i = \langle f_i, l_i, r_i \rangle \in \mathbb{A}$.
Output: A polynomial $\hat{g}(x_i) \in \mathbb{Z}[x_i]$, such that $ZERO(\hat{f}) \subseteq ZERO(\hat{g})$.
γ-g_1-g_2-list := $SIMPLE(\alpha_0, \alpha_1)$;
p-$\alpha_0(x) := g_1(x)$;
p-$\alpha_1(x) := g_2(x)$;
for $i = 2$ to n **do**
  $\alpha := \gamma$ {$\alpha = \langle a(x)$, a-l_i, a-$r_i \rangle$ }
  γ-g_1-g_2-list := $SIMPLE(\alpha, \alpha_i)$;
  p-$\alpha_i(x) := g_2(x)$;
  for $j = 0$ to $i - 1$ **do**
    p-$\alpha_j(x)$:= p-$\alpha_j(g_1(x))$; { Condition: $deg($ p-$\alpha_j(x)) < m = deg(a(x))$}
  end for
end for
$\alpha := \gamma$
p-$\alpha_i(x) := d \cdot$ p-$\alpha_i(x) \in \mathbb{Z}[x]$; {compute d for $0 \leq i \leq n$ such that ...}
$\hat{f}(x, y) := \sum_{j=0}^{n}$ p-$\alpha_j(x) y^j$;
$h(x) := GCD(a(x),$ p-$\alpha_n(x))$;
if $h(x) > 0$ **then**
  $a(x) := a(x)/h(x)$;
  p-$\alpha_i(x)$:= p-$\alpha_i(x)$ mod $a(x)$;
  p-$\alpha_i(x) := d \cdot$ p-$\alpha_i(x) \in \mathbb{Z}[x]$; {compute d for $0 \leq i \leq n$ such that ...}
end if
$\hat{g}(y) := RESULTANT_x(a(x), \hat{f}(x, y))$;
return $\hat{g}(y)$;

first component and a context in the second component. The context includes the free variables as well as the variables bound by real algebraic numbers in the second component, the real algebraic numbers of the extension and implicitely the order of the extension.

4.3 Assessments of the CAD Algorithm

Collins' cylindrical algebraic decomposition can solve the decision problem for a system of real non-linear multivariate polynomials in any real closed field for the following type of formulas:

$$\exists x_1 \cdots x_n (\bigwedge_{i=1}^{p} f_i(x_1, \cdots, x_n) \, \rho \, 0), \tag{4.17}$$

where $f_i \in \mathbb{R}[x_1, \cdots, x_n]$ and $\rho \in \{<, =, >\}$. Formulas of the form $\forall x(\phi)$ for any quantifier-free formula ϕ can be integrated via their equivalents $\neg\exists x(\neg\phi)$). Any polynomial inequation $f \neq 0$ is equivalent to $(f < 0) \vee (f > 0)$. The CAD algorithm only takes the set of polynomials $F = \{f_1, \cdots, f_p\}$ as input. It then returns a list of sample points, one for each cell of the $\mathcal{F}$-sign-invariant cylindrical algebraic decomposition of $\mathbb{R}^n$. In order to decide the consistency problem, each sample point has to be inserted into the set of polynomial equations and inequations until one sample point is found which satisfies the whole real polynomial system. If none of the sample points represents a solution of the polynomial system, it does not have a common real solution. Then, $SOL_{\mathbb{R}}(F) = \emptyset$.

The decision method using Collins' CAD algorithm is applicable for any ideal. This means, the ideal which is generated by the set of polynomials F does not have to be zero-dimensional. Parameters are handled like quantified variables in this method. The CAD algorithm is extendable to a general quantifier elimination (QE) algorithm, [Col98], which provides defining formulas for each valid cell.

Many improvements concerning efficiency aspects have been developed for the quantifier elimination method based on Collins' cylindrical algebraic decomposition. Improved projection operators are presented for example in [Hon98] and [McC98]. Especially, the partial cylindrical algebraic decomposition by Collins and Hong should be mentioned here, [CH98]. Hong implemented the partial cylindrical algebraic decomposition in a program called QEPCAD, which is available from him on request. It is based on the computer algebra C-library SACLIB. The algorithm is doubly exponential in the number of all variables, i.e. including parameters.

Considering the aspect of incrementality with respect to the CAD method used to decide the consistency problem, one has to confirm that the cylindrical algebraic decomposition of a consistent polynomial system cannot be reused in order to construct a cylindrical algebraic decomposition of the same consistent polynomial system extended by one additional polynomial equation or inequation. The cylindrical algebraic decomposition of the extended polynomial system has to be computed from scatch again.

Chapter 5

Practical Integration into the Racer System

One practical implementation of Buchberger's Algorithm is made available by Rychlik [Ryc00]. It is based on so-called Comprehensive Groebner Bases [Wei92]. In this appendix we present an examples that demonstrates how constraint systems with nonlinear equalities can be set up in Racer. For linear inequalities, Racer uses incremental constraint solving algorithms similar to those described in [JM94]. Thus, for specific classes of constraint systems, Racer uses more efficient algorithms. Before we can discuss the examples, some technical preliminaries must be presented.

5.1 Naming Conventions

Throughout this chapter we use the following abbreviations, possibly subscripted.

C	Concept term	*name*	Name of any sort
CN	Concept name	S	List of Assertions
IN	Individual name	GNL	List of group names
ON	Object name	LCN	List of concept names
R	Role term	*abox*	ABox object
RN	Role name	*tbox*	TBox object
AN	Attribute name	n	A natural number
ABN	ABox name	*real*	A real number
TBN	TBox name	*integer*	An integer number
KBN	knowledge base name	*string*	A string

5.2 RACER Knowledge Bases

In description logic systems a knowledge base is consisting of a TBox and an ABox. The conceptual knowledge is represented in the TBox and the knowledge about the instances of a domain is represented in the ABox.

5.2.1 Concept Language

The content of RACER TBoxes includes the conceptual modeling of concepts and roles as well. The modelling is based on the signature, which consists of two disjoint sets: the

$C \longrightarrow$ *CN* |
`*top*` |
`*bottom*` |
`(not` *C*`)` |
`(and` $C_1 \ldots C_n$`)` |
`(or` $C_1 \ldots C_n$`)` |
`(some` *R C*`)` |
`(all` *R C*`)` |
`(at-least` *n R*`)` |
`(at-most` *n R*`)` |
`(exactly` *n R*`)` |
`(at-least` *n R C*`)` |
`(at-most` *n R C*`)` |
`(exactly` *n R C*`)` |
`(a` *AN*`)` |
`(an` *AN*`)` |
`(no` *AN*`)` |
CDC

$R \longrightarrow$ *RN* |
`(inv` *RN*`)`

Figure 5.1: RACER concept and role terms.

set of concept names $\mathcal{C}$, also called the atomic concepts, and the set $\mathcal{R}$ containing the role names[1].

Starting from the set $\mathcal{C}$ complex concept terms can be build using several operators. An overview over all concept- and role-building operators is given in Figure 5.1.

Boolean terms build concepts by using the boolean operators.

	DL notation	RACER syntax
Negation	$\neg C$	`(not` *C*`)`
Conjunction	$C_1 \sqcap \ldots \sqcap C_n$	`(and` $C_1 \ldots C_n$`)`
Disjunction	$C_1 \sqcup \ldots \sqcup C_n$	`(or` $C_1 \ldots C_n$`)`

[1]The signature does not have to be specified explicitly in RACER knowledge bases - the system can compute it from the all the used names in the knowledge base - but specifying a signature may help avoiding errors caused by typos!

CDC $\longrightarrow$	`(min` *AN integer*`)`	|
	`(max` *AN integer*`)`	|
	`(equal` *AN integer*`)`	|
	`(equal` *AN AN*`)`	|
	`(divisible` *AN cardinal*`)`	|
	`(not-divisible` *AN cardinal*`)`	|
	`(>` *aexpr aexpr*`)`	|
	`(>=` *aexpr aexpr*`)`	|
	`(<` *aexpr aexpr*`)`	|
	`(<=` *aexpr aexpr*`)`	|
	`(<>` *aexpr aexpr*`)`	|
	`(=` *aexpr aexpr*`)`	|
	`(string=` *AN string*`)`	|
	`(string<>` *AN string*`)`	|
	`(string=` *AN AN*`)`	|
	`(string<>` *AN AN*`)`	
string $\longrightarrow$	`''` letter* `''`	
aexpr $\longrightarrow$	*AN*	|
	real	|
	`(+` *aexpr1 aexpr1**`)`	|
	aexpr1	

Figure 5.2: RACER concrete domain concepts and attribute expressions.

aexpr1 $\longrightarrow$		*aexpr2*	
	|	*aexpr3*	
	|	*aexpr5*	
aexpr2 $\longrightarrow$		*real*	
	|	*AN*	(AN of type real or complex)
	|	`(*` *real AN*`)`	(AN of type real or complex)
aexpr3 $\longrightarrow$		*real*	
	|	*AN*	(AN of type complex)
	|	`(*` *integer aexpr4 aexpr4**`)`	
aexpr4 $\longrightarrow$		*AN*	(AN of type complex)
	|	`(expt` *AN n*`)`	(AN of type complex)
aexpr5 $\longrightarrow$		*integer*	
	|	*AN*	(AN of type cardinal)
	|	`(*` *integer AN*`)`	(AN of type cardinal)

Figure 5.3: Specific expressions for predicates ($n > 0$ is a natural number) .

Qualified restrictions state that role fillers have to be of a certain concept. Value restrictions assure that the type of *all* role fillers is of the specified concept, while exist restrictions require that there be *a* filler of that role which is an instance of the specified concept.

	DL notation	RACER syntax
Exists restriction	$\exists\ R.C$	`(some` R C`)`
Value restriction	$\forall\ R.C$	`(all` R C`)`

Number restrictions can specify a lower bound, an upper bound or an exact number for the amount of role fillers each instance of this concept has for a certain role. Only roles that are not transitive and do not have any transitive subroles are allowed in number restrictions (see also the comments in [HST00]).

	DL notation	RACER syntax
At-most restriction	$\leq n\ R$	`(at-most` n R`)`
At-least restriction	$\geq n\ R$	`(at-least` n R`)`
Exactly restriction	$= n\ R$	`(exactly` n R`)`
Qualified at-most restriction	$\leq n\ R.C$	`(at-most` n R C`)`
Qualified at-least restriction	$\geq n\ R.C$	`(at-least` n R C`)`
Qualified exactly restriction	$= n\ R.C$	`(exactly` n R C`)`

Actually, the exactly restriction `(exactly` n R`)` is an abbreviation for the concept term `(and (at-least` n R`) (at-most` n R`))` and `(exactly` n R C`)` is an abbreviation for the concept term `(and (at-least` n R C`) (at-most` n R C`))`

There are two concepts implicitly declared in every TBox: the concept "top" ($\top$) denotes the top-most concept in the hierarchy and the concept "bottom" ($\bot$) denotes the inconsistent concept, which is a subconcept to all other concepts. Note that $\top$ ($\bot$) can also be expressed as $C \sqcup \neg C$ ($C \sqcap \neg C$). In RACER $\top$ is denoted as `*top*` and $\bot$ is denoted as `*bottom*`[2].

[2]For KRSS compatibility reasons RACER also supports the synonym concepts `top` and `bottom`.

Concrete domain concepts state concrete predicate restrictions for attribute fillers (see Figure 5.2.1). RACER currently supports three unary predicates for integer attributes (`min`, `max`, `equal`), six nary predicates for real attributes (`>`, `>=`, `<`, `<=`, `=`, `<>`), a unary existential predicate with two syntactical variants (`a` or `an`), and a special predicate restriction disallowing a concrete domain filler (`no`). The restrictions for attributes of type `real` have to be in the form of linear inequations (with order relations) where the attribute names play the role of variables. If an expression is built with the rule for *aexpr*4 (see Figure 5.2.1), a so-called nonlinear constraint is specified. In this case, only equations and inequations (`=`, `<>`), but no order constraints (`>`, `>=`, `<`, `<=`) are allowed, and the attributes must be of type `complex`. If an expression is built with the rule for *aexpr*5 (see Figure 5.2.1) a so-called cardinal linear constraint is specified, i.e., attributes are constrainted to be a natural number (including zero). Racer also supports a concrete domain for representing equations about strings with predicaes `string=` and `string<>`. The use of concepts with concrete domain expressions is illustrated with examples in Section 5.3. For the declaration of types for attributes, see see Section 5.2.4.

	DL notation	RACER syntax
Concrete filler exists restriction	$\exists A.\top_{\mathcal{D}}$	`(a` *A*`)` or `(an` *A*`)`
No concrete filler restriction	$\forall A.\bot_{\mathcal{D}}$	`(no` *A*`)`
Integer predicate exists restriction with $z \in \mathbb{Z}$	$\exists A.min_z$	`(min A` *z*`)`
	$\exists A.max_z$	`(max A` *z*`)`
	$\exists A.=_z$	`(equal A` *z*`)`
Real predicate exists restriction with $P \in \{>, >=, <, <=, =\}$	$\exists A_1, \ldots, A_n.P$	`(`*P aexpr aexpr*`)`

An all restriction of the form $\forall A_1, \ldots, A_n.P$ is currently not directly supported. However, it can be expressed as disjunction: $\forall A_1.\bot_{\mathcal{D}} \sqcup \cdots \sqcup \forall A_n.\bot_{\mathcal{D}} \sqcup \exists A_1, \ldots, A_n.P$.

5.2.2 Concept Axioms and Terminology

RACER supports several kinds of concept axioms.

General concept inclusions (GCIs) state the subsumption relation between two concept terms.
DL notation: $C_1 \sqsubseteq C_2$
RACER syntax: `(implies` C_1 C_2`)`

Concept equations state the equivalence between two concept terms.
DL notation: $C_1 \doteq C_2$
RACER syntax: `(equivalent` C_1 C_2`)`

Concept disjointness axioms state pairwise disjointness between several concepts. Disjoint concepts do not have instances in common.
DL notation: $C_1 \sqsubseteq \neg(C_2 \sqcup C_3 \sqcup \cdots \sqcap C_n)$
$C_2 \sqsubseteq \neg(C_3 \sqcup \cdots \sqcap C_n)$
$\ldots$
$C_{n-1} \sqsubseteq \neg C_n$
RACER syntax: `(disjoint` C_1 `...`C_n`)`

Actually, a concept equation $C_1 \doteq C_2$ can be expressed by the two GCIs: $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. The disjointness of the concepts $C_1 \ldots C_n$ can also be expressed by GCIs.

There are also separate forms for concept axioms with just concept names on their left-hand sides. These concept axioms implement special kinds of GCIs and concept equations. But concept names are only a special kind of concept terms, so these forms are just syntactic sugar. They are added to the RACER system for historical reasons and for compatibility with KRSS. These concept axioms are:

Primitive concept axioms state the subsumption relation between a concept name and a concept term.
DL notation: $(CN \sqsubseteq C)$
RACER syntax: `(define-primitive-concept` *CN C*`)`

Concept definitions state the equality between a concept name and a concept term.
DL notation: $(CN \doteq C)$
RACER syntax: `(define-concept` *CN C*`)`

Concept axioms may be cyclic in RACER. There may also be forward references to concepts which will be "introduced" with `define-concept` or `define-primitive-concept` in subsequent axioms. The terminology of a RACER TBox may also contain several axioms for a single concept. So if a second axiom about the same concept is given, it is added and does not overwrite the first axiom.

5.2.3 Role Declarations

In contrast to concept axioms, role declarations are unique in RACER. There exists just one declaration per role name in a knowledge base. If a second declaration for a role is given, an error is signaled. If no signature is specified, undeclared roles are assumed to be neither a feature nor a transitive role and they do not have any superroles.

The set of all roles ($\mathcal{R}$) includes the set of features ($\mathcal{F}$) and the set of transitive roles ($\mathcal{R}^+$). The sets $\mathcal{F}$ and $\mathcal{R}^+$ are disjoint. All roles in a TBox may also be arranged in a role hierarchy. The inverse of a role name *RN* can be either explicitly declared via the keyword `:inverse` (e.g. see the description of `define-primitive-role` in Section **??**, page **??**) or referred to as `(inv` *RN*`)`.

Features (also called attributes) restrict a role to be a functional role, e.g. each individual can only have up to one filler for this role.

Transitive Roles are transitively closed roles. If two pairs of individuals IN_1 and IN_2 and IN_2 and IN_3 are related via a transitive role R, then IN_1 and IN_3 are also related via R.

Role Hierarchies define super- and subrole-relationships between roles. If R_1 is a superrole of R_2, then for all pairs of individuals between which R_2 holds, R_1 must hold too.

In the current implementation the specified superrole relations may not be cyclic. If a role has a superrole, its properties are not in every case inherited by the subrole. The properties of a declared role induced by its superrole are shown in Figure 5.4. The table

		Superrole $RN_1 \in$		
		$\mathcal{R}$	$\mathcal{R}^+$	$\mathcal{F}$
Subrole RN_1	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{R}$	$\mathcal{F}$
declared as	$\mathcal{R}^+$	$\mathcal{R}^+$	$\mathcal{R}^+$	-
element of:	$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$	$\mathcal{F}$

Figure 5.4: Conflicting declared and inherited role properties.

should be read as follows: For example if a role RN_1 is declared as a simple role and it has a feature RN_2 as a superrole, then RN_1 will be a feature itself.

The combination of a feature having a transitive superrole is not allowed and features cannot be transitive. Note that transitive roles and roles with transitive subroles may not be used in number restrictions.

RACER does not support role terms as specified in the KRSS. However, a role being the conjunction of other roles can as well be expressed by using the role hierarchy. The KRSS-like declaration of the role `(define-primitive-role` RN `(and` RN_1 RN_2`))` can be approximated in RACER by: `(define-primitive-role` RN `:parents (`RN_1 RN_2`))`.

KRSS	DL notation
`(define-primitive-role` RN `(domain` C`))`	$(\exists\ RN.\top) \sqsubseteq C$
`(define-primitive-role` RN `(range` D`))`	$\top \sqsubseteq (\forall\ RN.D)$
RACER Syntax	DL notation
`(define-primitive-role` RN `:domain` C`)`	$(\exists\ RN.\top) \sqsubseteq C$
`(define-primitive-role` RN `:range` D`)`	$\top \sqsubseteq (\forall\ RN.D)$

Figure 5.5: Domain and range restrictions expressed via GCIs.

RACER offers the declaration of domain and range restrictions for roles. These restrictions for primitive roles can be either expressed with GCIs, see the examples in Figure 5.5 or declared via the keywords `:domain` and `:range`.

5.2.4 Concrete Domain Attributes

RACER supports reasoning over natural numbers ($\mathbb{N}$), integers ($\mathbb{Z}$), reals ($\mathbb{R}$), complex numbers ($\mathbb{C}$), and strings. For different sets, different kinds of predicates are supported.

$\mathbb{N}$	linear inequations with order constraints and integer coefficients
$\mathbb{Z}$	interval constraints
$\mathbb{R}$	linear inequations with order constraints and rational coefficients
$\mathbb{C}$	nonlinear multivariate inequations with integer coefficients
Strings	equality and inequality

For the users convenience, rational coefficients can be specified in floating point notation. They are automatically transformed into their rational equivalents (e.g., 0.75 is transformed into 3/4). In the following we will use the names on the left-hand side of the table to refer to the correspondings concrete domains.

Names for values from concrete domains are called *objects*. The set of all objects is referred to as $\mathcal{O}$. Individuals can be associated with objects via so-called *attributes names*

(or attributes for short). Note that the set $\mathcal{A}$ of all attributes must be disjoint to the set of roles (and the set of features). Attributes can be declared in the signature of a TBox (see below).

Attributes are considered as "typed" since they can either have fillers of type `cardinal`, `integer`, `real`, `complex`, or `string`. The same attribute cannot be used in the same TBox such that both types are applicable, e.g., `(min has-age 18)` and `(>= has-age 18)` are not allowed. If the type of an attribute is not explicitly declared, its type is implicitly derived from its use in a TBox/ABox. An attribute and its type can be declared with the signature form or by using the KRSS-like form `define-concrete-domain-attribute`. If an attribute is declared to be of type `complex` is can be used in linear (in-)equations. However, if an attribute is declare to be of type `real` or `integer` it is an error to use this attribute in terms for nonlinear polynoms. In a similar way, currently, an attribute of type `integer` may not be used in a term for a linear polynoms, either. If the coefficients are integers, then `cardinal` (natural number, including 0) for the type of attributes may be used in a linear polynom. Furthermore, attributes of type `string` may not be used on polynoms, and non-strings may not be used in constraints for strings.

5.2.5 ABox Assertions

An ABox contains assertions about individuals. The set of individual names (or individuals for brevity) $\mathcal{I}$ is the signature of the ABox. The set of individuals must be disjoint to the set of concept names and the set of role names. There are four kinds of assertions:

Concept assertions with `instance` state that an individual IN is an instance of a specified concept C.

Role assertions with `related` state that an individual IN_1 is a role filler for a role R with respect to an individual IN_2.

Attribute assertions with `constrained` state that an object ON is a filler for a role R with respect to an individual IN.

Constraints within `constraints` state relationships between objects of the concrete domain. The syntax for constraints is explained in Figure 5.2.1. Instead of attribute names, object names must be used.

5.3 An Example for Linear Inequalities

The following example is an extension of the family TBox introduced above. In the example, the concrete domains $\mathbb{Z}$ and $\mathbb{R}$ are used.

```
...
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes ((integer age)))
...
(equivalent teenager (and human (min age 16)))
(equivalent old-teenager (and human (min age 18)))
...
```

Asking for the children of teenager reveals that `old-teenager` is a `teenager`. A further extensions demonstrates the usage of reals as concrete domain.

```
...
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes ((integer age)
                (real temperature-celsius)
                (real temperature-fahrenheit)))
...
(equivalent teenager (and human (min age 16)))
(equivalent old-teenager (and human (min age 18)))
(equivalent human-with-feaver (and human (>= temperature-celsius 38.5))
(equivalent seriously-ill-human (and human (>= temperature-celsius 42.0)))
...
```

Obviously, RACER determines that the concept `seriously-ill-human` is subsumed by `human-with-feaver`. For the reals, RACER supports linear equations and inequations. Thus, we could add the following statement to the knowledge base in order to make sure the relations between the two attributes `temperature-fahrenheit` and `temperature-celsius` is properly represented.

```
(implies top (= temperature-fahrenheit
                (+ (* 1.8 temperature-celsius) 32)))
```

If a concept `seriously-ill-human-1` is defined as

```
(equivalent seriously-ill-human-1
            (and human (>= temperature-fahrenheit 107.6)))
```

RACER recognizes the subsumption relationship with `human-with-feaver` and the synonym relationship with `seriously-ill-human`.

In an ABox, it is possible to set up constraints between individuals. This is illustrated with the following extended ABox.

```
...
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes (...)
   :individuals (eve doris)
   :objects (temp-eve temp-doris))
...
(constrained eve temp-eve temperature-fahrenheit)
(constrained doris temp-doris temperature-celsius)
(constraints
   (= temp-eve 102.56)
   (= temp-doris 39.5))
```

For instance, this states that `eve` is related via the attribute `temperature-fahrenheit` to the object `temp-eve`. The initial constraint `(= temp-eve 102.56)` specifies that the object `temp-eve` is equal to 102.56.

Now, asking for the direct types of eve and doris reveals that both individuals are instances of `human-with-feaver`. In the following Abox there is an inconsistency since the temperature of 102.56 Fahrenheit is identical with 39.5 Celsius.

```
(constrained eve temp-eve temperature-fahrenheit)
(constrained doris temp-doris temperature-celsius)
(constraints
   (= temp-eve 102.56)
   (= temp-doris 39.5)
   (> temp-eve temp-doris))
```

We present another example that might be important for many applications: dealing with dates. The following declarations can be processed with Racer. The predicates `divisible` and `not-divisible` are defined for natural numbers and are reduced to linear inequations internally.

```
(define-concrete-domain-attribute year :type cardinal)
(define-concrete-domain-attribute days-in-month :type cardinal)

(implies Month (and (>= days-in-month 28) (<= days-in-month 31)))

(equivalent month-inleapyear
            (and Month
                 (divisible year 4)
                 (or (not-divisible year 100)
                     (divisible year 400))))

(equivalent February
            (and Month
                 (<= days-in-month 29)
                 (or (not month-inleapyear)
                     (= days-in-month 29))
                 (or month-inleapyear
                     (= days-in-month 28))))
```

Next, we assume some instances of February are declared.

```
(instance feb-2003 February)
(constrained feb-2003 year-1 year)
(constrained feb-2003 days-in-feb-2003 days-in-month)
(constraints (= year-1 2003))

(instance feb-2000 February)
(constrained feb-2000 year-2 year)
(constrained feb-2000 days-in-feb-2000 days-in-month)
(constraints (= year-2 2000))
```

Note that the number of days for both months is not given explicitly. Nevertheless, asking `(concept-instances month-inleapyear)` yields `(feb-2000)` whereas asking for `(concept-instances (not month-inleapyear))` returns `(feb-2003)`. In addition, one could check the number of days:

```
(constraint-entailed? (<> days-in-feb-2003 29))
(constraint-entailed? (= days-in-feb-2000 29))
```

In both cases, the answer is true.

5.4 An Example with Nonlinear Equations

The following examples how easily nonlinear constraints can be set up with Racer.

```
(in-knowledge-base test)

(define-concrete-domain-attribute x :type complex)
(define-concrete-domain-attribute y :type complex)

(implies human-on-circle
         (= (+ (* 1 x x) (* 1 y y) -1) 0))

(implies human-on-hyperbola
         (= (+ (* 1 x x x) (* 1 y y)) 0))

(instance lara human-on-circle)
(constrained lara x-pos-lara x)
(constrained lara y-pos-lare y)

(instance leon human-on-hyperbola)
(constrained leon x-pos-leon x)
(constrained leon y-pos-leon y)

(constraints (= x-pos-lara x-pos-leon))
(constraints (= y-pos-lara y-pos-leon))
```

If Racer is asked whether the knowledge base is consistent, the answer is true. However, currently, Racer does not support nonlinear inequations (based on order relations), so it is not possible to specify that Lara's location is somewhere IN the circle (as indicated in the example in the introdcution). Furthermore, the range of the concrete domain attributes used in the example must be declared as `complex`. As indicated in this report, real root counting algorithms must be integrated into Racer to support also real attributes in nonlinear (in-)equations. In the current version, real attributes can only be used in linear inequations. If concrete solutions for constraint systems are required, algorithms for cylindrical algebraic decomposition might be implemented in future versions of Racer. As long as constraint systems are underconstrained, it might be possible to compute minimal interval for (some) variables in constraint systems. This kind of inference service might also be supported in a future version of Racer.

Chapter 6

Comparison

This work presents the decision problem for real closed fields with respect to the application in description logics. Different methods using Gröbner bases computation or applying cylindrical algebraic decomposition are explained and analyzed to solve the decision problem. In Figure 6.1, the discussed methods are depicted in an overview. Additionally, two methods are included which have not been considered in this work but should be mentioned as well. Weispfenning introduces a quantifier elimination (QE) method based on virtual term substitution, [Wei97]. This method is doubly exponential in the number of quantifier blocks but only singly exponential in the number of quantified variables, [Dol99]. The quantifier elimination method using virtual term substitution is implemented in a REDUCE-package, called REDLOG, whose source code and documentation is freely available in the internet. REDLOG also provides interfaces to QEPCAD, Hong's implementation of the partial cylindrical algebraic decomposition, and to QERRC, the QE method using real root counting and comprehensive Gröbner bases. The second method is presented by Basu, which states a better theoretical complexity than a doubly exponential complexity, [Bas99]. This quantifier elimination method has not been implemented yet, but seems to be interesting for further investigations.

The Gröbner basis methods, as presented in Chapter 3, can be applied to solve the decision problem for real closed fields with respect to the application in description logics. The advantage of these methods is that an incremental version of Buchberger's algorithm for generating a Gröbner basis can be implemented. The main disadvantage poses the fact that the dimension of the solution set is restricted to zero. This means that all kinds of problems which have an infinite number of solutions cannot be applied to these methods. The extensions of the basic Gröbner basis method with the Elimination Method or the Univariate Polynomial Method have even restrictions on the form of the input equations and inequations such that ordering inequations are not allowed. Thus, among the methods based on Gröbner basis, the the Gröbner basis thechnique combined with real root counting is the only method that allows polynomial equations and inequations with ordering relations. Therefore, it can be applied to solve the decision problem for an application such as in a concrete domain for the description logic (DL) system RACERas illustrated in the introduction.

The method based on Collins' cylindrical algebraic decomposition can also be used to solve the the decision problem for real closed fields with respect to the application in

description logics. Here, the advantage is given by the applicability to all polynomial systems including those which have an infinite solution space. Moreover, equations and inequations containing ordering inequations are allowed for input. However, this method is not adaptable to an incremental version, which should be mentioned as a disadvantage with respect to the application in description logics.

One of the main problems in symbolic-algebraic computations is due to storage limitations and not due to time limitiations. In most cases, the size of the expressions arising during the computations exceed the storage capacity. Research studies for the common real zeros of a polynomial system can not yet present general solving strategies for a whole class of polynomial systems. Moreover, it also seems to be difficult to determine the class of polynomials for which a certain quantifier elimination method is best applicable. Therefore, it is rather important to combine and explore the different advantages of several quantifier eliminations methods to solve a problem. In [DSW97] and in [Dol99], a comparison between QERRC, QEPCAD and REDLOG for some practical examples has been evaluated. As a conclusion, it cannot be stated that one of the three methods is superior to the others. In [Hon91], Hong compares three QE algorithms with respect to their complexities on the decision problem: Collins' cylindrical algebraic decompostion method, Grigor'ev and Vorobjov's method (1988) and Renegar's method (1989). It turned out that comparing theoretical complexities can lead to wrong estimations on the average case since theoretical complexities consider worst case scenarios. Even though Renegar's algorithm should be faster than Grigor'ev 's which should be faster than Collins', it seems to be the case that Collins' algorithm is the fastest with respect to average problems.

In general, it is more complicated to find the common zeros of a system of polynomials over fields which are not algebraically closed. Concerning future investigations, it seems to be of great importance to introduce solving methods which directly search for the real solutions without having to compute the complex roots as well since there exist many problems where the number of real zeros compose only a small fraction of the number of total zeros [Pet97]. Thus, we hope that this work provides a good basis for ongoing implementations to solve the decision problem for real closed fields in many different applications such as in a concrete domain for the description logic (DL) system RACER.

Figure 6.1: Methods for the Decision Problem

Bibliography

[ACM98] D.S. Arnon, G.E. Collins, and S. McCallum. Cylindrical algebraic decomposition I: The basic algorithm. In Caviness and Johnson [CJ98], pages 136–151.

[Bas99] S. Basu. New results on quantifier elimination over real closed fields and applications to constraint databases. *Journal of the ACM*, 46(4):537–555, 1999.

[BCL82] B. Buchberger, G.E. Collins, and R. Loos, editors. *Computer Algebra. Symbolic and Algebraic Computation*, volume 4 of *Computing Supplementum*. Springer-Verlag, 1982.

[BOKR86] M. Ben-Or, D. Kozen, and J. Reif. The complexity of elementary algebra and geometry. *Journal of Computer and System Sciences*, 32:251–264, 1986.

[Buc85] B. Buchberger. Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory. In N. K. Bose, editor, *Multidimensional Systems Theory*, pages 184–232. D. Reidel Publishing Company, 1985.

[Buc89] B. Buchberger. Applications of Gröbner bases in non-linear computational geometry. In Kapur and Mundy [KM89], pages 413–446.

[CH98] G.E. Collins and Hoon Hong. Partial cylindrical algebraic decomposition for quantifier elimination. In Caviness and Johnson [CJ98], pages 174–200.

[CJ98] B.F. Caviness and Jeremy R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer-Verlag, 1998.

[CLO96] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, 2 edition, 1996.

[Col98] G.E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In Caviness and Johnson [CJ98], pages 85–121.

[CR88] M. Coste and M. F. Roy. Thom's lemma, the coding of real algebraic numbers and the computation of the topology of semi-algebraic sets. *Journal of Symbolic Computation*, 5:121–129, 1988.

[CY94] S. Chakrabarti and K. Yelick. Distributed data structures and algorithms for Gröbner basis computation. *Lisp and Symbolic Computation: An international Journal*, 7:147–172, 1994.

[Dol94] A. Dolzmann. Reelle Quantorenelimination durch parametrisches Zählen von Nullstellen. Diploma thesis, Universität Passau, D-94030 Passau, Germany, November 1994.

[Dol99] A. Dolzmann. Solving geometric problems with real quantifier elimination. Technical Report MIP-9903, Fachbereich Mathematik und Informatik, Universität Passau, D-94030 Passau, Germany, 1999.

[DSW97] A. Dolzmann, Th. Sturm, and V. Weispfenning. Real quantifier elimination in practice. Technical Report MIP-9720, Fachbereich Mathematik und Informatik, Universität Passau, D-94030 Passau, Germany, 1997.

[Fis86] G. Fischer. *Lineare Algebra*. Vieweg Verlag, 1986.

[HM02] V. Haarslev and R. Möller. Practical Reasoning in RACER with a Concrete Domain. In *Proceedings of the 2002 Intl. Workshop on Description Logics (DL2002)*, pages 1–7, 2002.

[HMW01] V. Haarslev, R. Möller, and M. Wessel. The description logic $\mathcal{ALCNH}_{R+}$ extended with concrete domains: A practically motivated approach. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy*, LNCS, pages 29–44. Springer-Verlag, New York, Berlin, Heidelberg, June 2001.

[Hon91] H. Hong. Comparison of several decision algorithms for the existential theory of the reals. *Journal of*, pages 1–33, 1991.

[Hon98] H. Hong. An improvement of the projection operator in cylindrical algebraic decomposition. In Caviness and Johnson [CJ98], pages 166–173.

[HST00] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic SHIQ. In D. MacAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482–496. Springer-Verlag, 2000.

[Jir95] M. Jirstrand. Cylindrical algebraic decomposition - an introduction. Technical report, Automatic Control group, Department of Electrical Engineering, Linköping University, S-58183 Linköping, Sweden, 1995.

[JM94] J. Jaffar and M. Maher. Constraint logic programming: a survey. *The Journal of Logic Programming*, 19-20:1–29, 19994.

[Kap86] Deepak Kapur. Using Gröbner Bases to reason about geometry problems. *Journal of Symbolic Computation*, 2:399–408, 1986.

[KM89] D. Kapur and J.L. Mundy, editors. *Geometric Reasoning*. The MIT Press, 1989.

[KS86] B. Kutzler and S. Stifter. On the application of Buchberger's algorithm to automated geometry theorem proving. *Journal of Symbolic Computation*, 2:389–397, 1986.

[Lip93] F. Lippold. Implementierung eines Verfahrens zum Zählen reeller Nullstellen multivariater Polynome. Diploma thesis, Universität Passau, D-94030 Passau, Germany, September 1993.

[Loo82] R. Loos. Computing in algebraic extensions. In Buchberger et al. [BCL82], pages 173–187.

[McC98] S. McCallum. An improved projection operator for cylindrical algebraic decomposition. In Caviness and Johnson [CJ98], pages 242–268.

[Mis93] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, 1993.

[Mon96] E. Monfroy. *Solver Collaboration for Constraint Logic Programming*. PhD thesis, Université Henri Poincaré - Nancy I, November 1996.

[Pet97] S. Petitjean. Algebraic geometry and computer vision: Polynomial systems, real and complex roots. *Journal of Mathematical Imaging and Vision*, pages 1–32, 1997.

[PRS93] P. Pedersen, M.-F. Roy, and A. Szpirglas. Counting real zeros in the multivariate case. In F. Eyssette and A. Galligo, editors, *Computational Algebraic Geometry (Proceedings of MEGA'92)*, pages 203–224. Birkhäuser, 1993.

[Ryc00] M. Rychlik. Complexity and applications of parametric algorithms of computational algebraic geometry. In *R. del la Llave, L. Petzold, and J. Lorenz, editors,* Dynamics of Algorithms*, volume 118 of The IMA Volumes in Mathematics and its Applications*. Springer Verlag, 2000.

[Sei01] A. Seidl. Effiziente Realisierung reeller algebraischer Zahlen. Diploma thesis, Universität Passau, D-94030 Passau, Germany, August 2001.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[Tra00] Q.-N. Tran. A fast algorithm for Gröbner basis conversion and its applications. *Journal of Symbolic Computation*, 30(4):451–467, 2000.

[Wei92] V. Weispfenning. Comprehensive Groebner bases. *Journal of Symbolic Computation*, 14:1–29, 1992.

[Wei93] V. Weispfenning. A new approach to quantifier elimination for real algebra. Technical Report MIP-9305, Fachbereich Mathematik und Informatik, Universität Passau, D-94030 Passau, Germany, 1993.

[Wei97] V. Weispfenning. Quantifier elimination for real algebra - the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, 1997.

[Wei98] V. Weispfenning. A new approach to quantifier elimination for real algebra. In Caviness and Johnson [CJ98], pages 376–392.

[Yap00] C.K. Yap. *Fundamental Problems of Algorithmic Algebra*. Oxford University Press, 2000.