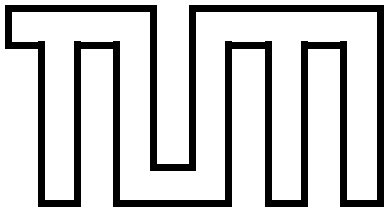

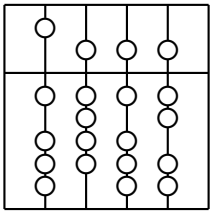

Automated methods for formal proofs in simple arithmetics and algebra

Amine Chaieb

Lehrstuhl für Software & Systems Engineering
Institut für Informatik
Technische Universität München

Lehrstuhl für Software & Systems Engineering
Institut für Informatik
Technische Universität München

Automated methods for formal proofs in simple arithmetics and algebra

Amine Chaieb

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Hans-Joachim Bungartz

Prüfer der Dissertation: 1. Univ.-Prof. Tobias Nipkow, Ph.D.

2. Ass.Prof. Jeremy Avigad, Ph.D.
Canergie Mellon University, Pittsburgh/USA

Die Dissertation wurde am 31. Januar 2008 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 01. April 2008 angenommen.

Zusammenfassung

In einem LCF-ähnlichen Theorembeweiser, stammt jeder Beweis aus einer minimalen Menge von Inferenzregeln ab. Somit sind Verfahren zur Generierung solcher Beweise von enormer Wichtigkeit. Das Ziel dieser Abhandlung ist folgende Frage zu studieren: *Wie soll, allgemein und im Spezialfall der Arithmetik, ein LCF-ähnlicher Theorembeweiser um eine Entscheidungsprozedur erweitert werden?*. Wir betrachten drei verschiedene Ansätze für eine solche Integration und präsentieren mehrere Beweisverfahren im Detail. Die wichtigsten präsentierten Verfahren sind: a) Entscheidungsprozeduren für universelle und schwach existentielle Probleme in Ringen, b) Univerelle Probleme reeller Polynome, c) Quantorenelimination für parametrische lineare Formeln über geordnete Körper, Presburger Arithmetik, die gemischte lineare Theorie der reelen und ganzen Zahlen, Algebraisch- und Reellabgeschlossene Körper. Alle unsere Arbeiten basieren auf dem Isabelle Theorembeweiser.

Abstract

In an LCF-like theorem prover, any proof must be produced from a small set of inference rules. The development of automated proof methods in such systems is extremely important. In this thesis we study the following question *How should we integrate a proof procedure in an LCF-like theorem prover, both in general and in the special case of arithmetics.* We investigate three integration paradigms and present several proof procedures. These include universal and weak existential problems over rings, universal polynomial problems over the reals, quantifier elimination for parametric linear problems over ordered fields, Presburger arithmetic, mixed real-integer linear arithmetic, algebraically and real closed fields. Our work has been carried out in the Isabelle framework.

To Moufida, Salem and Rayfa.

I dreamed of a tree of indescribable beauty and great height. Growing on it were three kinds of fruit unlike any fruit known in this world. The size of a girl's breast, each shone like a moon or a sun against the green surface of the tree. Awestruck, I looked at the marvellous tree and asked, "Whence comes this tree?"

Acknowledgements

I am very grateful to Tobias Nipkow for giving me the opportunity to work in his group and learn about theorem proving. His tactical advices, persuasions for interesting topics and dissuasions from dangerous roads were of invaluable help.

I am very grateful to Jeremy Avigad, who had astonishing readiness to answer my questions and give lucid explanations. I thank him for his constructive discussions, motivating comments and for giving me the honour to be my second adviser and examiner.

Great distraction is mandatory to oversee John Harrison's influence on this work. I am indebted to him for very constructive and encouraging discussions, advices and for giving me access to preliminary versions of his book.

I am grateful to all my (ex-)colleagues for the nice atmosphere at the research group. Stefan Berghofer, Sascha Böhme, Florian Haftmann, Alexander Kraus, Walid Maalej, Norbert Schirmer and Makarius Wenzel made substantial comments on early drafts. I thank Norbert and Sascha for the agreeable atmosphere at the office and for their patience with my loud music. I thank Stefan and Makarius for various technical support and patience with my requirements. I have learned from them that building and maintaining a working system is by far the hardest part of theorem proving.

I can hardly thank my parents and my adorable wife enough for their constant love, support and all the joyful moments. I am indebted to them for two of my sources of happiness. I owe my education entirely to the great sacrifices of my selfless parents, and the existence and happiness of our own little family to the devotion of my good-hearted wife.

Contents

Chapter 1

Introduction

> Was beweisbar ist, soll in der Wissenschaft
> nicht ohne Beweis geglaubt werden.
>
> *(J. W. R. Dedekind)*

One major contribution of symbolic logic is to clarify the notion of a *mathematical proof*. There, we use a fixed formal language and a set of inference rules for deduction. Any proof must be expressed by a finite composition of the inference rules. Of course the used symbols have a mathematical meaning, and the resulting deduction calculus must be at least consistent. In practise, the formal language is represented on a computer and the applicability of an inference rule is decidable. This thesis should be understood in the context of this latter practical side of symbolic logic: theorem proving.

The pioneer system LCF [GMW79] provided general design principles for theorem provers, which have been embraced by the HOL family [HSA06], Coq [TLG06] and the Isabelle/Isar framework [WP06, Wen07]. All these systems, like all theorem provers, provide procedures to automatically generate a *formal proof* for suitable classes of problems. Proving even simple tautologies formally by hand is involved and time consuming. The above systems provide proof-procedures e.g. for simplification, propositional and predicate logic and simple arithmetics. Although arithmetic plays a central role in mathematics, modelling systems, verification and engineering problems, proof automation for it is seldom satisfactory. Non-linear statements are rarely proved automatically and even for linear arithmetics most systems cannot deal with quantifiers.

The goal of this thesis is to study the following question:

> *How should we integrate proof-procedures into an LCF-like system both in general and for particular relevant theories of arithmetic?*

To study the integration issues of proof procedures *in general*, we consider the LCF architecture more closely. The main design principle in the LCF system is to couple a theorem prover with a full programming language (the *meta language* or ML for short) and exploit its strongly typed environment. In ML, theorems are values of an abstract type, whose constructors implement the logical inferences. The most natural way to integrate a proof procedure is maybe by implementing it in ML. For every problem instance, it goes through the inference rules in a sophisticated manner to prove the goal. We shall refer to such an integration as a *derived rule*, since the new procedure can be seen as a new inference rule justifying its results in terms of other inference rules. A second approach is to find a *certificate*, which yields a simple proof of the goal, if interpreted appropriately. The certificate can be found by any software solving a problem inspired by the goal. A very simple example is that 7 is a certificate to show that 3 divides 21, since $3 \cdot 7 = 21$. We refer to this approach as *certificate-based*. The last approach we will consider in this thesis is based on the following observation. The object language (HOL) contains a functional programming language with a big advantage over ML: we can prove our programs correct in HOL. We refer to this approach of implementing the procedures *inside* the logic by *reflection*.

To study the integration issues of proof procedures *for particular relevant theories of arithmetic*, we give a relatively broad case study applying the different approaches to some non-trivial theories including universal and existential polynomial problems, quantifier elimina-

tion procedures for linear arithmetic over $\mathbb{R}$, $\mathbb{Z}$ and their mixed theory, and for algebraically- and real-closed fields. All our work has been done in Isabelle/HOL [NPW02].

Contributions

My main contributions in this thesis are as follows:

1. A broad case study on three paradigms for the integration of proof-procedures in an LCF like theorem prover.
2. Various concrete proof methods (§2.1.3, §4.2 and §3.2) to work with respect to logical contexts and abstract specification mechanisms (locales and type-classes [KWP99, Bal04, Bal06, HW06]).
3. An abstract and executable formalisation of univariate and multivariate polynomials with several utilities.
4. Reflected formalisations of full quantifier elimination for linear arithmetic over $\mathbb{Z}$ and $\mathbb{R}$, their mixed theories and for parametric linear arithmetic over ordered fields.
5. First steps towards clarifying the usage of virtual substitution to prove and implement quantifier elimination procedures in Isabelle. Our use of "improper terms" as locale parameters to lift a simple quantifier elimination procedure for dense linear orders without endpoints to one for linear arithmetic over ordered fields is new.
6. Implementation of proof-procedures and formal proofs for non-trivial theorems, e.g. the fundamental theorem of algebra in §4.4.

Overview

This thesis is structured as follows. In chapter 2 we address the issues of integrating proof procedures and present the three paradigms sketched above in more detail with an elaborated example for each approach. Chapter 2 is hence dedicated to study the *general* part of our main question. We study particular cases in the rest of the thesis. Chapter 3 is dedicated to the application of the certificate-based approach to simple polynomial problems: universal and a subset of $\forall\exists$ problems over rings and universal problems over $\mathbb{R}$ with ordering. In chapter 4, we focus on the method of quantifier elimination, which has inherently bad certificates. We present, in derived rule and reflection style, quantifier elimination procedures for dense linear orders, linear arithmetics (over $\mathbb{R}$, $\mathbb{Z}$ and their mixed theory) and the elementary theories of real and algebraically closed fields. We draw some conclusions in Chapter 5.

Notation

In this thesis Mathematics, ML, and HOL are distinguished as three different worlds, where the notions of theorems and functions have different meanings.

In Mathematics, theorems and helpful lemmas are proved on paper. We shall present these as in Theorem 1 and (1.a), though with more formal notation.

Any composite number is measured by some prime number. (1.a)

Theorem 1 (Euclid)**.** *Prime numbers are more than any assigned multitude of prime numbers.*

Important is that lemmas (e.g. (1.a)) will be labelled *alphabetically*, in contrast to HOL theorems, which will be labelled by *numbers*. A function f from A to B will be denoted by the usual arrow notation: $f : A \to B, \quad x \mapsto f(x)$.

The meta-language ML is the programming language in which the theorem prover is implemented, i.e. SML in Isabelle's case. We will *not* present SML-code but rather a self explanatory pseudo-code, and this in very few places only. An ML function f from type α to type β will be denoted by $f : \alpha \to \beta$. In particular abstraction in ML is realised by a *bold* $\boldsymbol{\lambda}$, in contrast to the normal λ for HOL functions. We assume pattern matching over terms with guards like in Haskell. A theorem in ML is an element of an abstract

type, whose constructors implement the underlying logic. Matching a theorem statement against a pattern is possible using the **as**'...' notation. If *th* is a theorem whose *statement* is $x > 0 \to x^2 > 0$ then *th* **as**'$A \to B$' instantiates A to $x > 0$ and B to $x^2 > 0$. If p is a theorem for $x > 0$, then *fwd th p* yields a theorem for $x^2 > 0$. The instantiation of the variables $x_1, \ldots, x_n$ *in the order of their appearance* in a theorem q is realised by $q[x_1, \ldots, x_n]$, i.e. the statement of $th[0]$ is $0 > 0 \to 0^2 > 0$. This view of ML is not unrealistic, see [WC07].

In HOL we can see theorems as abstract entities with names and we will present them as in (1.1).

$$\forall x, y :: \mathbb{Z}. x < y \to 2 \cdot x + 1 < 2 \cdot y \tag{1.1}$$

Note that these in contrast to mathematical statements are labelled with *numbers*, unlike the real theory sources in Isabelle. We assume that the presented theorems are accessible in ML using their labels. *All HOL theorems presented in this thesis have been formally proved in Isabelle.* A HOL function f from HOL type α to HOL type β will be denoted by $f :: \alpha \Rightarrow \beta$. *All* HOL *functions are total.* We declare datatypes in HOL using datatype and write functions by pattern matching. The HOL type $[\alpha]$ denotes lists of elements of type α. These are constructed from the empty list [] and consing $x \cdot xs$. We write $[x_1, \ldots, x_n]$ (*also with the informal dots*) for clarity reasons. We denote the set of all elements of a list xs by $\{\!\{xs\}\!\}$, the n^{th} element of xs by $xs!n$ and appending two lists xs and ys by $xs@ys$. The HOL type $\alpha \times \beta$ represents all pairs (a, b) where $a :: \alpha$ and $b :: \beta$. Many of the HOL theorems we will present are not "formally" correct as presented. For example, we shall write

$$(P \leftrightarrow P') \wedge (Q \leftrightarrow Q') \to (P \bowtie Q) \leftrightarrow (P' \bowtie Q'), \quad \text{for } \bowtie \in \{\wedge, \vee, \to, \leftrightarrow\} \tag{1.2}$$

as a meta-representation of a family of theorems. We will use, for the three worlds, $\emptyset, \mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ and $\mathbb{C}$ to denote the empty set, natural number, integers, rational, real and complex numbers respectively. In this thesis $0 \in \mathbb{N}$. The HOL type $[\mathbb{R}]$ is for instance the type of all lists over the reals. We will use this notation also for ML. The $\in$ symbol denotes set membership in each world. Also arithmetical operations $+, \cdot, /$ and $-$ denote addition, multiplication, division, and (overloaded) negation and subtraction. Power will be denoted by an exponent. The same applies to the logical connectives $\wedge, \vee, \to, \leftrightarrow$, quantifiers $\forall$ and $\exists$, function composition $\circ$, list operations, and equality $=$, heavily overloaded in types and worlds.

We hope the reader agrees that this simplifies the notation and makes the material more accessible.

Informal notes

Using theorem provers which examine your arguments with minuteness like Isabelle is not always easy. In my experience, descriptions of formalisations fall into two categories: those full of *moan*, inciting pity and presenting the task as extremely hard and painful; and those focusing on the content of the work describing it as *simple*, although not true in particular cases. I write this thesis in the second style. I am aware that some parts might, on that account, sound bold or unworthy. I hazard this risk, so much I dislike the first style. I believe that presenting things as simple and abstracted from personal judgements is much more convenient to the reader, who should concentrate on the *content* of the work. Personal judgements differ with persons, systems and time.

Related work

The content of this thesis lies in the intersection of several interesting fields, e.g. symbolic logic and computation, algebra, automated reasoning, theorem proving and computational complexity. Due to this diversity we found it much more appropriate to discuss related work at the end of every chapter. These discussions are by no means exhaustive, but include all the references we used in our study and suggestions for further reading.

Chapter 2

Integration of proof-procedures

Contents

In this chapter, we study the integration issues of new proof-procedures. We concentrate the study on three paradigms: a) derived rules (in §2.1) where we implement the procedure in ML and invoke a sequence of inference-rules to prove the goal, b) reflection (in §2.2) where we implement the procedure *inside* the logic and prove it correct once and for all, and c) certificate-based methods (in §2.3) where we search, using external software, for a succinct certificate, yielding a proof of the goal, when appropriately checked. We present simple examples to illustrate each of the approaches. These three paradigms, especially a) and b), have several advantages and drawbacks and raise quite challenging logical problems. We close the chapter by a short comparison of the methods.

2.1 Derived rules

Rules in the LCF approach are the constructors of the abstract type *thm* of theorems. They represent the inference rules of the implemented logic and live in the critical module we call "kernel". Let in the following "rule" denote any ML function whose range type is *thm*. We call rules different from the kernel rules *derived rules*, since they construct their results by sooner or later calling the kernel-rules. Implementing derived rules in practice requires intimate knowledge of system internals. This integration paradigm has so far been the most popular and various derived rules are available in the system, as well as generic mechanisms to manipulate rules. We sketch some of these generic mechanisms in §2.1.1. In §2.1.2 we give an overview of the Isar architecture. We emphasise the presentation on the basic notion of contexts, which allows to integrate proof-procedure that depend on a logical context. Using this last methodology, we come closest to meeting the expectations of mathematicians. For instance, a decision procedure for the word problem in rings, should work for all rings, regardless of the underlying set or how the operations are named. We conclude by an example presenting a quantifier elimination procedure for dense linear orders without endpoints.

2.1.1 Basic tools for LCF proof composition

We give a rough idea about the tool chest available in the pure approach.

Conversions: combinators for equational reasoning

A conversion *cv* is a special kind of rules with signature $term \to thm$. The implicit contract is that $cv\,t$ returns a theorem $t = t'$. Conversions are a powerful tool for equational reasoning,

e.g the Isabelle simplifier is a rather complex conversion. Note that we can scrutinise the input t and prove $t = t'$, with t' in the exact syntactical shape we need. Thereby, the whole theorem-proving machinery available so far is at hand.

Typically, we write conversions for special purposes and then combine them to more complex ones using so called conversionals. The conversional *argc*, for instance, applies a conversion to t when called on term $f\ t$, hence returning $f\ t = f\ t'$. Similarly *func* applies a conversion cv to f in the example above, *absc* applies cv to the body of $\lambda x.t$, and *binopc* applies a conversion to both arguments of a binary operation. Given two conversions cv_1 and cv_2 then the conversion cv_1 *thenc* cv_2 applied to t returns $t = t''$, where $t = t'$ and $t' = t''$ are the theorems returned by $cv_1\ t$ and $cv_2\ t'$ respectively. We use these conversions in §2.1.3. Other conversionals include applying several conversions in a specific order or applying a given conversion to all sub-terms of a term etc. We can consider conversionals as a tiny programming language to write conversions.

Tactics: generic background reasoning

In Isabelle, a tactic is any ML function taking a theorem representing a goal state and returning a sequence of theorems representing possible follow-on goal states. The theorem proving task may hence be "defined" as emptying that resulting list of pending goals. Tactics are often special purpose, e.g. predicate logic, arithmetic, rewriting, etc. They are therefore best suited to prove theorems on the fly needed inside a derived rule. We can build more sophisticated tactics from simpler ones by means of tacticals. These relate to tactics in a similar way conversionals relate to conversions. Tacticals can hence be seen as a simple language for writing tactics. Prominent tacticals include sequential composition, trying one or several tactics, repeating a tactic, applying a conversion to a goal and many others.

Generic theorem synthesis

The following special combinator plays an important role in practice and in many developments in this thesis:

thm-of decomp t =
  let (*ts*, *recomb*) = *decomp t*
  in *recomb* (*map* (*thm-of decomp*) *ts*)

It takes a problem decomposition function $decomp : \alpha \to [\alpha] \times ([\beta] \to \beta)$ and a problem $t : \alpha$, decomposes t into a list of sub-problems ts and a recombination function *recomb*, solves the sub-problems recursively, and combines their solution into an overall solution via *recomb*. Often problems are formulae and solutions are theorems. This style of theorem proving originates from the LCF system [GMW79, Pau87], where *decomp* is called a *tactic* — not to be confused with our tactics. The underlying principle is divide and conquer. This technique is especially helpful to construct induction proofs, where problems are decomposed in the induction case and recombined with the induction hypothesis. We use this technique extensively in §4.2.

All presented techniques are very appealing since they allow one to combine the different proof-tools available in the system into special purpose ones in a *uniform* manner: they operate on the universal term data structure and are hence applicable in a wide range of problems.

2.1.2 Higher concepts in the Isar framework

From the tool-builder's perspective, Isar is *roughly* the main organiser of Isabelle's theorem-proving machinery. We present only the few features we need later, but see [Wen02, WP06, Wen07, CW07] for more details.

Generic contexts and context data In primitive inferences $\Gamma \vdash A$ means that A is derivable within a context Γ, which contains arbitrary logical and extra-logical data, e.g. back-

ground theory declarations (types, constants, axioms), local parameters and assumptions, definitions, theorems, syntax and type-inference information, hints for proof tools (e.g. simplification rules) etc. There are two main logical operations on contexts:

1. *Context construction* starts with an empty context Γ_0 and proceeds by adding further context elements consecutively. In particular, $\Gamma + \mathbf{fix}\ x$ declares a local variable, and $\Gamma + \mathbf{assume}\ A$ states a local assumption.
2. *Context export* destructs the logical difference of two contexts, by imposing it on local results. The effect of *export* $\Gamma_1\ \Gamma_2$ is to discharge portions of the context on terms and theorems as follows:

$$\begin{array}{lcl} \mathit{export}\ (\Gamma + \mathbf{fix}\ x)\ \Gamma\ (t\ x) & = & \lambda x.t\ x \\ \mathit{export}\ (\Gamma + \mathbf{fix}\ x)\ \Gamma\ (\vdash Bx) & = & (\vdash\ \bigwedge x.B\ x) \\ \mathit{export}\ (\Gamma + \mathbf{assume}\ A)\ \Gamma\ (A \vdash B) & = & (\vdash A \Longrightarrow B) \end{array}$$

Internally, context data consists of an inhomogeneous record of individual data slots, based on dynamically typed disjoint sums. The external programming interface recovers strong static typing by means of an SML functor, involving dependently-typed modules, *functor Data(ARGS): RESULT*, where:

$$\begin{array}{rcl} \mathit{ARGS} & = & \mathit{sig\ type\ T\ val\ init:\ T\ end} \\ \mathit{RESULT} & = & \mathit{sig\ val\ put:\ T \to\ context\ \to\ context} \\ & & \quad \mathit{val\ get:\ context \to T} \end{array}$$

We emphasise that the ML type *context* assimilates *all* ML types. This means that we can attach *arbitrary* ML code to contexts. We use this feature to attach proof procedures using the local assumptions and theorems in a context.

Morphisms Morphisms organise certain logical operations by determining how results may be transferred from one context into another, providing a different *view*. Formally, a morphism ϕ is just an ML-mapping on theorems. ϕ_0 refers to the identity morphism. The following two kinds of morphisms, which resemble abstraction and application in λ-calculus, are particularly important in practice.

1. *Export morphism:* the export operation between two contexts determines a morphism $\phi = \mathit{export}\ \Gamma_1\ \Gamma_2$, giving a generalised view of local results.
2. *Interpretation morphism:* given concrete terms for the fixed variables, and theorems for the assumptions of a context, the substitution operation determines a morphism $\phi = \mathit{interpret}\ [t/x]\ [th/A]$. By this view, results of an abstract theory are turned into concrete instances.

Generic declarations Since morphisms are mappings on theorems, their application to the logical part of a context is immediate. We handle the application of morphisms to arbitrary data at the level of *data declarations*. Recall that arbitrary datatypes can be incorporated into the generic *context* and hence any operation on data is subsumed by *context* $\to$ *context*. This motivates the following definition:

$$\mathit{declaration} = \mathit{morphism}\ \to \mathit{context} \to \mathit{context}$$

This means a declaration participates in applying the morphism. Being passed some ϕ as additional argument it is supposed to apply it to any logical parameters (types, terms, theorems) involved in its operation. Note that immediate declaration in the current context works by passing the identity morphism ϕ_0.

A traditional *fact declaration* in Isabelle is represented as a theorem-attribute pair. This is an important special case. The general declaration is recovered by the *apply* operation:

$$\begin{array}{rcl} \mathit{fact\_declaration} & = & \mathit{thm} \times \mathit{attribute} \\ \mathit{attribute} & = & \mathit{thm} \to \mathit{context} \to \mathit{context} \\ \mathit{apply}\ (\mathit{th}, \mathit{att}) & = & \boldsymbol{\lambda}\phi.\mathit{att}\ (\phi\ \mathit{th}) \end{array}$$

Observe that *th* is transformed separately before invoking the *attribute*. Here *att* does not have to consider morphisms at all. A prominent example is the declaration of simplification rules.

Locales and type-classes *Locales* [KWP99] provide a high-level mechanism to organise context elements and declarations. *Locale expressions* [Bal04] compose existing locales by means of merge and rename operations. *Locale interpretation* [Bal06] transfers results stemming from a locale into another context. *Type-classes* [Wen97] express properties of polymorphic entities within the type-system. There is a canonical interpretation of classes as specific locales [HW06]. All locale operations are reduced to basic inferences usually expressed via morphisms. A locale specification consists of the following two distinctive parts.

1. *Assumptions* refer to fixed types, terms, and hypotheses, specified by "**fixes**" and "**assumes**". The definition "**locale** c = **fixes** x **assumes** $A\ x$" produces a context construction $\Gamma +$ **fix** x **assume** $A\ x$, and a predicate constant $c = \lambda x.A\ x$.
2. *Conclusions* are essentially theorems or definitions that depend on the locale context. We denote these using the notation "**(in** c**)**" as in (2.1) and (2.2).

$$\textbf{fun (in } c\textbf{) } \mathsf{foo}\ a = \textbf{if } A\ a \textbf{ then } a \textbf{ else } x \tag{2.1}$$

$$\textbf{(in } c\textbf{) } \forall y.A(\mathsf{foo}\ y) \tag{2.2}$$

The conclusion part is what really matters in practical use, including declarations of arbitrary extra-logical data. Locales maintain a canonical order of declarations $d_1, \ldots, d_n$, used to reconstruct the context relative to a given morphism ϕ. The context is augmented by the collective declaration of $d_n\ \phi\ (\cdots(d_1\ \phi\ \Gamma)\cdots)$. Thus we may attach arbitrary SML values to a locale, which will be transformed together with the logical content by morphism application.

2.1.3 Example: dense linear orders

In this example we develop a quantifier elimination procedure (qep. for short) for the first order theory of dense linear orders without endpoints (DLO). Quantifier elimination (qe.) asks for a qep. which given any DLO formula ϕ returns a quantifier-free (qf.) formula ψ, which is equivalent to ϕ in the DLO theory. Langford [Lan27] proved DLO to admit qe. and we develop a proof-producing version of his algorithm. We axiomatise DLO as follows:

$$\begin{aligned}&\textbf{locale } dlo = linorder\ +\ \textbf{assumes}\\ &\qquad \forall x.\exists u.x < u \wedge \exists l.l < x \wedge \forall x, y.x < y \rightarrow \exists z.x < z < y\end{aligned}$$

Generic qe.

Assume we have a function *qe* to eliminate one existential quantifier over a quantifier free formula, i.e. *qe* $(\exists x.P(x))$ yields a theorem for $(\exists x.P(x)) \leftrightarrow Q$ for qf. P and Q. Then we can eliminate all quantifiers in a formula by applying *qe* recursively, see function *liftqe* below. Note that *qe* is a conversion and hence *liftqe* is a conversional:

liftqe qe P =
case P **of**
  $\neg A \Rightarrow$ *argc* (*liftqe qe*) P
  $A \beta B \mid \beta \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \Rightarrow$ *binopc* (*liftqe qe*) P
  $\forall x.A \Rightarrow$ *fwd trans* [*all_ex*, *liftqe qe* $\neg\exists x.\neg A$]
  $\exists x.A \Rightarrow$ ((*argc* $\circ$ *absc* $\circ$ *liftqe qe*) *thenc qe*) P
  $\_ \Rightarrow$ *refl*[P]

The theorems *all_ex*, *trans* and *refl* denote $\forall x.P(x) \leftrightarrow \neg\exists x.\neg P(x)$, transitivity and reflexivity of $\leftrightarrow$ respectively. The interesting case is when P has the form $\exists x.A$. There we first "convert" the body A and eliminate all its quantifiers. Note that $\exists x.A(x)$ is internally the constant $\exists$ applied to $\lambda x.A(x)$ and hence the use of *argc* and *absc*. We assume terms to

be in η-expanded form just for the presentation. After this step the theorem at hand is $\exists x.A(x) \leftrightarrow \exists x.A'(x)$, where A' is qf. Therefore we just need to apply *qe* to $\exists x.A'(x)$.

In the actual implementation, the *liftqe* conversional takes other conversions to be applied at specific steps of the qe., e.g. normalise atomic formulae or simplify the formula before and after applying *qe*. This allows several optimisations such as distributing $\exists$ over $\vee$. These conversions and *qe* itself also depend on an environment of free variables and variables bound by outer quantifiers. Note that by this mechanism, *qe* can safely assume its input to have a specific shape, e.g. it is in negational normal form (NNF) and all atoms are of a specific syntactical shape imposed by the variables.

In the following we develop *langford*, an instance of *qe* for DLO.

A proof-procedure for Langford's algorithm

The first step is to transform the input into disjunctive normal form (DNF). We assume that the input is a formula $\exists x.P(x)$, where P is a conjunction of atoms $x < y$ or $z < x$, for y and z different from x. A proof-producing version of this step is simple and omitted. Now consider a formula Q corresponding to $\exists x. \bigwedge_{i=1}^{n} l_i < x \wedge \bigwedge_{i=1}^{m} x < u_j$. The goal is to produce a proof that this is equivalent to $\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{m} l_i < u_j$, if $n > 0$ and $m > 0$, and to *True* if $n = 0$ or $m = 0$. First transform Q equivalently to $\exists x.(\forall l \in L.l < x) \wedge (\forall u \in U.x < u)$ for suitable HOL sets L and U. This step formalises the previous informal notation of $\bigwedge_{i=1}^{n} \ldots$. We abuse notation here and write $x < S$ (resp. $S < x$) for $\mathsf{finite}\ S \wedge \forall s \in S.x < s$ (resp. $\mathsf{finite}\ S \wedge \forall s \in S.s < x$) and $S < S'$ for $\forall x \in S. \forall y \in S'.x < y$. For the transformation, we automatically prove the trivial properties (2.3)–(2.10). Given $\exists x.P(x)$, we instantiate (2.3) and then rewrite with (2.4)–(2.10).

$$\textbf{(in } dlo\textbf{)}: \ (\exists x.P(x)) \leftrightarrow (\exists x.\emptyset < x \wedge x < \emptyset \wedge P(x)) \tag{2.3}$$

$$\textbf{(in } dlo\textbf{)}: \ (\exists x.L < x < U \wedge x < u \wedge P(x)) \leftrightarrow (\exists x.L < x < \{u\} \cup U \wedge P(x)) \tag{2.4}$$

$$\textbf{(in } dlo\textbf{)}: \ (\exists x.L < x < U \wedge l < x \wedge P(x)) \leftrightarrow (\exists x.\{l\} \cup L < x < U \wedge P(x)) \tag{2.5}$$

$$\textbf{(in } dlo\textbf{)}: \ (\exists x.L < x < U \wedge x < u) \leftrightarrow (\exists x.L < x < \{u\} \cup U) \tag{2.6}$$

$$\textbf{(in } dlo\textbf{)}: \ (\exists x.L < x < U \wedge l < x) \leftrightarrow (\exists x.\{l\} \cup L < x < U) \tag{2.7}$$

Now we can apply the appropriate instance of (2.8), (2.9) or (2.10). Note that the sets L and U are proved finite by construction.

$$\textbf{(in } dlo\textbf{)}: \ L \neq \emptyset \wedge U \neq \emptyset \wedge \mathsf{finite}\ L \wedge \mathsf{finite}\ U \rightarrow (\exists x.L < x < U) \leftrightarrow L < U \tag{2.8}$$

$$\textbf{(in } dlo\textbf{)}: \ L \neq \emptyset \wedge \mathsf{finite}\ L \rightarrow (\exists x.L < x < \emptyset) \leftrightarrow \mathit{True} \tag{2.9}$$

$$\textbf{(in } dlo\textbf{)}: \ U \neq \emptyset \wedge \mathsf{finite}\ U \rightarrow (\exists x.\emptyset < x < U) \leftrightarrow \mathit{True} \tag{2.10}$$

Integration as a context-aware method

We exhibit an integration of the previous qep. working in different contexts, i.e. inside locales. We refer to such procedures as *context-aware* or *context-sensitive*. Clearly the previous procedure only depends on the theorems (2.3)–(2.10). The integrated method reserves a data slot in the generic Isar-context to manage the different instances of *dlo*. This is a mapping from keys to entries: keys are theorems identifying the locale instance, and entries consist of the *transformed* theorems (2.3)–(2.10). Keys are technically just the locale predicate instantiated according to the interpretation.

The main part of the integration code is the ML structure *LangfordData* below. To add the theorems automatically for every instance, we use the fact declaration below.

$$\begin{array}{lcl} \mathit{type\ entry} & = & \{\mathit{qes} : [\mathit{thm}],\ \mathit{gs} : [\mathit{thm}],\ \mathit{atoms} : [\mathit{term}]\} \\ \mathit{type\ key} & = & \mathit{thm} \\ \mathit{structure\ LangfordData} & = & \mathit{Data}(\mathit{type\ T} = [\mathit{key} * \mathit{entry}]) \end{array}$$

declare(in *dlo*) *dlo.axioms*[*langford qes*: (2.8)–(2.10) *gs*: (2.3)–(2.7) *atoms:* $< \leq$]

$$\mathsf{datatype}\ \sigma = \widehat{\mathbb{R}} \mid \boldsymbol{v}_{\mathbb{N}} \mid \sigma + \sigma \mid \sigma - \sigma \mid \sigma * \sigma \mid \sigma^{\mathbb{N}} \mid \sum_{\mathbb{N}}^{\mathbb{N}} \sigma$$

$$\begin{array}{lll} (\!|t + s|\!)^e_\sigma & = (\!|t|\!)^e_\sigma + (\!|s|\!)^e_\sigma & (\!|\widehat{c}|\!)^e_\sigma = c \\ (\!|t - s|\!)^e_\sigma & = (\!|t|\!)^e_\sigma - (\!|s|\!)^e_\sigma & (\!|\boldsymbol{v}_n|\!)^e_\sigma = \underline{e!n} \\ (\!|t * s|\!)^e_\sigma & = (\!|t|\!)^e_\sigma \cdot (\!|s|\!)^e_\sigma & (\!|t^{\boldsymbol{n}}|\!)^e_\sigma = ((\!|t|\!)^e_\sigma)^n \\ & & (\!|\sum_n^m t|\!)^e_\sigma = \sum_{k=n}^m (\!|t|\!)^{k\cdot e}_\sigma \end{array}$$

Figure 2.1: Syntax and semantics of σ-terms

Here *langford* is an Isar-attribute which parses *and checks* the arguments *qes, gs* and *atoms* into the context data. The procedure performs qe. as explained above using only these theorems, obtained by *LangfordData.get*, see §2.1.2. When the method is invoked on a goal P, we extract an atom from P and search for a matching instance installed in the context. For this we just match the extracted atom against the atoms in the context data. When we find a corresponding instance, we call the core-method with the corresponding entries. The formalisation of this example took around 100 lines of Isabelle proofs and 300 lines of SML code.

2.2 Reflection

> Or, as we recognise the reflection of letters in the water, or in a mirror, only when we know the letters themselves
>
> *(Plato, The Republic)*

Reflection means to perform a proof-step by computation *inside* the logic. Assume, for concreteness, that we want to prove $t = t'$ for two terms $t :: \tau$ and $t' :: \tau$, where t' is algorithmically obtained from the *structure* of t. A problem arises when the structure of τ-terms is not accessible inside the logic, since two syntactically different τ-terms may be logically equivalent. For that consider formulae or integer expressions. Evidently this systematic transformation and its correctness proof, cannot be defined inside the logic.

The basic idea of reflection is to represent, i.e. *reflect*, the relevant subset of τ-terms inside the logic using a datatype *rep* (called shadow syntax in [Har96, Har95]) and define the transformation $simp :: rep \Rightarrow rep$ as a recursive function. To ensure correctness and integration we define the semantics of *rep*-terms $(\!|\cdot|\!) :: rep \Rightarrow \tau$ and prove the theorem $(\!|simp\ s|\!) = (\!|s|\!)$. Now given a τ-term t, a proof of $t = t'$ needs two steps.

Reification: conjure up a *rep*-term s and prove $(\!|s|\!) = t$. This is an ML function.

Evaluation: instantiate *simp*'s correctness theorem above, compute $s' = simp\ s$ and obtain the theorem $(\!|s'|\!) = (\!|s|\!)$. The final result holds by transitivity.

The evaluation step is crucial for efficiency and can be performed in several manners using: a) rewriting, b) execution of generated ML code or c) an internal λ-calculus evaluator. We discuss the impacts of the evaluation approaches in §2.4.

2.2.1 Example: a simple case of sums

We reflect the computation of very simple sums: the bounds and the index variable are natural numbers and the sum body is a real polynomial in the index variable. The syntax (datatype σ) and its semantics in Figure 2.2.1 reflect the considered terms. We represent variables by de Brujin indices: $\boldsymbol{v}_n$ represents the bound variable with index $n :: \mathbb{N}$. The semantics $(\!|.|\!)_\sigma$ is parametrised by an environment $e :: [\mathbb{N}]$. Note that $(\!|\boldsymbol{v}_n|\!)^e_\sigma$ is the real number injected from the natural number $e!n$, the n^{th} element in e. The bold symbols $+$, $*$ etc are constructors and reflect their counterparts $+$, $\cdot$ etc in the logic. In particular, a coefficient $c :: \mathbb{R}$ is represented by $\widehat{c}$.

$$\begin{array}{ll}
\mathsf{selim}\ se\ \sum_n^m t & = \textbf{if}\ n \le m\ \textbf{then}\ se\ n\ m\ (\mathsf{selim}\ se\ t)\ \textbf{else}\ \widehat{0} \\
\mathsf{selim}\ se\ (t \circ s) & = (\mathsf{selim}\ se\ t) \circ (\mathsf{selim}\ se\ s) \quad \text{for } \circ \in \{+, -, *\} \\
\mathsf{selim}\ se\ t^{\boldsymbol{n}} & = (\mathsf{selim}\ se\ t)^{\boldsymbol{n}} \\
\mathsf{selim}\ se\ t & = t
\end{array}$$

Figure 2.2: Generic elimination of $\sum$ in σ-terms

Generic $\sum$-elimination

Let $\mathsf{sfree}\ t$ formalise that the σ-term t contains no $\sum$ and let se be a function that eliminates a $\sum$ in front of any $\sum$-free σ-term. The algorithm selim of Figure 2.2.1 eliminates all the $\sum$ symbols. Isabelle can then prove (2.11) automatically. Hence, we only need to provide an instance of se for eliminating the $\sum$ in $\sum_n^m t$, where $\mathsf{sfree}\ t$ holds.

$$\begin{array}{r}
(\forall e, n, m, s.\mathsf{sfree}\ s \wedge n \le m \rightarrow (\!|se\ n\ m\ s|\!)_\sigma^e = (\!|\sum_n^m s|\!)_\sigma^e \wedge \mathsf{sfree}(se\ s)) \\
\rightarrow \forall e, t.(\!|\mathsf{selim}\ se\ t|\!)_\sigma^e = (\!|t|\!)_\sigma^e \wedge \mathsf{sfree}(\mathsf{selim}\ se\ t)
\end{array} \tag{2.11}$$

For $t :: \sigma$, $\mathsf{unbound}\ t$ formalises that t does not contain $\boldsymbol{v}_0$. Function isnorm_σ imposes a normal form on σ-terms. Moreover we assume without loss of generality that t is in the normal form defined below, i.e. $\mathsf{isnorm}_\sigma\ t$. Any $\sum$-free σ-term t can be transformed into a σ-term in normal form using norm_σ, simple and omitted, cf. (2.12).

$$\begin{array}{ll}
\mathsf{isnorm}_\sigma\ (c * \boldsymbol{v}_0{}^{\boldsymbol{m}} + t) & = \mathsf{unbound}\ c \wedge \mathsf{isnorm}_\sigma\ t \\
\mathsf{isnorm}_\sigma\ t & = \mathsf{unbound}\ t
\end{array}$$

$$\mathsf{sfree}\ t \rightarrow (\!|\mathsf{norm}_\sigma\ t|\!)_\sigma^e = (\!|t|\!)_\sigma^e \wedge \mathsf{isnorm}_\sigma\ (\mathsf{norm}_\sigma\ t) \tag{2.12}$$

Differential algebra

Summation plays a similar role in the discrete case as integration in the continuous one. The dual of derivative in this context is the difference operator Δ defined by: $\Delta\ f = \lambda x.f(x+1) - f(x)$. It is easy to prove the following for Δ:

$$\Delta\ f = g \rightarrow \sum_{k=n}^{m} g(k) = f(m+1) - f(n). \tag{2.13}$$

Hence computing a sum over g reduces to computing f, an indefinite sum of g often denoted by $f = \sum g$, such that $\Delta\ f = g$. In our case we have to compute $\sum_{k=n}^m \underline{k}^l$, due to the normal form. For this we only need to find a function f such that $\Delta\ f = \lambda k.\underline{k}^l$. In the continuous case this would have been $\lambda x.\frac{x^{l+1}}{l+1}$. In the discrete case this property is not preserved for "normal" powers, but for "falling" powers. The n^{th} falling power of x is denoted by $x^{\underline{n}}$ and formalises $\prod_{i=0}^{n-1} x - i$, cf. (2.14). Property (2.15) states that the indefinite sum of $\lambda x.x^{\underline{k}}$ is $\lambda x.\frac{x^{\underline{k+1}}}{k+1}$.

$$x^{\underline{0}} = 1 \quad | \quad x^{\underline{n+1}} = x \cdot (x-1)^{\underline{n}} \tag{2.14}$$

$$\Delta(\lambda x.\frac{x^{\underline{k+1}}}{k+1}) = \lambda x.x^{\underline{k}} \tag{2.15}$$

In order to convert between normal and falling powers, we use Stirling numbers, cf. (2.16). We have $\sum_{k=n}^m \underline{k}^l = \sum_{k=n}^m \sum_{i=0}^l \mathsf{S}_{l,i} \cdot \underline{k}^i = \sum_{i=0}^l \mathsf{S}_{l,i} \cdot \sum_{k=n}^m \underline{k}^i = \sum_{i=0}^l \frac{1}{i+1} \cdot \mathsf{S}_{l,i} \cdot ((m+1)^{\underline{i+1}} - n^{\underline{i+1}})$. Now we can convert normal powers to falling powers using (2.17).

$$\mathsf{S}_{0,0} = 1 \quad | \quad \mathsf{S}_{0,k+1} = \mathsf{S}_{n+1,0} = 0 \quad | \quad \mathsf{S}_{n+1,k+1} = (k+1) \cdot \mathsf{S}_{n,k+1} + \mathsf{S}_{n,k} \tag{2.16}$$

$$x^n = \sum_{k=0}^{n} \mathsf{S}_{n,k} \cdot x^{\underline{k}} \tag{2.17}$$

We implement function se to compute the result as above and to decrease the de Brujin indices, using decr_σ, since one $\sum$ has been eliminated. The function $(\mathsf{se} \circ \mathsf{norm}_\sigma)$ satisfies the premise of (2.11) and finally (2.18) holds. The formalisation of this example took 1100 lines of Isabelle proofs.

$$\begin{aligned}
\mathsf{se}\ m\ n\ (d * \boldsymbol{v}_0{}^{\boldsymbol{k}} + t) &= \textbf{let}\ c = \mathsf{foldl}\ (\lambda s, i.s + ((m+1)^{\underline{i+1}} - n^{\underline{i+1}}) \cdot \tfrac{\mathsf{S}_{k,i}}{i+1})\ 0\ [0..k] \\
&\quad\ \textbf{in}\ (\mathsf{decr}_\sigma\ d) * \widehat{c} + \mathsf{se}\ n\ m\ t \\
\mathsf{se}\ n\ m\ t &= (\mathsf{decr}_\sigma\ t) * \widehat{m - n + 1}
\end{aligned}$$

$$(\!|\mathsf{selim}\ \mathsf{se}\ t|\!)^e_\sigma = (\!|t|\!)^e_\sigma \wedge \mathsf{sfree}(\mathsf{selim}\ \mathsf{se}\ t). \tag{2.18}$$

2.2.2 Generic reification as derived rule

Remember that to integrate a reflected proof procedure, two steps are needed: reification and evaluation. Evaluation is automatic. We present in the following an algorithm to automate reification in several practical cases. Consider a term t and a set of theorems (E_i) below, and let $\overline{e_i}$ be short for $e_i^1\ \ldots\ e_i^{n_i}$, where $1 \le i \le k$.

$$\begin{aligned}
f_1\ t_1\ e_1^1\ \ldots\ e_1^{n_1} &= P_1(r_1, \ldots, r_{m_1}) \quad (E_1) \\
\vdots \qquad\qquad\quad &= \\
f_k\ t_k\ e_k^1\ \ldots\ e_k^{n_k} &= P_k(r_k, \ldots, r_{m_k}) \quad (E_k),
\end{aligned}$$

The goal is to find t's representation, i.e. a term s, and environments $\overline{e_j}$ and *prove* $t = f_j\ s\ \overline{e_j}$. A simple example is to take the equations in Figure 2.2.1, where $f_i = \lambda t, e.(\!|t|\!)^e_\sigma$. The following algorithm works for several interdependent interpretation functions. We assume the following restrictions:

1. f_i is a rigid term of type $\rho_i \Rightarrow \gamma_1 \Rightarrow \cdots \Rightarrow \gamma_{n_1} \Rightarrow \tau_i$ for all $1 \le i \le k$ and $\tau_i = \tau_j$ implies $f_i = f_j$,
2. the patterns t_i and P_i are rigid patterns,
3. the r_i represent the only occurrences of f_i in P_i,
4. binding in the shadow syntax occurs only via de Brujin indices and through environments (of HOL type $[\alpha]$, for some α) using consing, cf. the summation example in §2.2.1.

The basic idea is that reification consists of rewriting with the inverted equations above. Of course this does not allow to discover the environments. In order to apply the techniques of §2.1.1, we transform each of the E_i's into a congruence rule C_i expressing the decomposition into smaller reification problems:

$$\bigwedge_{k=1}^{m_i} x_k = r_k \to P_i(x_1, \ldots, x_{m_i}) = f_i\ t_i\ \overline{e_i} \quad (C_i),\ \text{for fresh } x_1, \ldots, x_{m_i}$$

These rules are trivial to prove. In order to deal with bindings, we introduce environments (ML lists) Γ_α and Θ_α for free and bound environment variables, for every $e_i^j :: [\alpha]$. The algorithm proceeds as follows:

R_1: Find C_i such that $P_i(x_i, \ldots, x_{m_i})$ matches t with θ as a *mgu* and let $C_i' = \theta C_i$ be $\bigwedge_{k=1}^{m_i} s_k = r_k' \to t = f_i\ t_i'\ \overline{e_i}$. Then decompose the problem into $[s_1, \ldots, s_{m_i}]$ and recombine with *fwd* C_i'. If no such C_i exists go to step R_2.

R_2: We assume $t = \lambda(x :: \alpha).t'$, otherwise go to step R_3. Add x to Θ_α and decompose the problem into $[t']$. For recombination generalise over x, use extensionality and remove x from Θ_α.

R_3: Find E_i such that $P_i(y_1, \ldots, y_{l_i})$ matches t with θ as a most general unifier. Here $y_1, \ldots, y_{l_i}$ are fresh variables replacing any occurrence of $e_i^j!n$ for any j and n. Add $s_j = \theta(y_j)$ to Γ_{α_j} where s_j has type α_j, for $1 \le j \le l$. Let k_j be the index of s_j in $\Theta_{\alpha_j}@\Gamma_{\alpha_j}$. Decompose into the empty set of problems, and recombine by returning $\theta(E_i)$ instantiated as follows: $n_j \mapsto k_j, e_i^j \mapsto \overline{y} \cdot d_i^j$, for every $1 \le j \le l$, where $\Theta_{\alpha_j} = [\overline{y}]$ and $d_i^j, 1 \le i \le k, 1 \le j \le l$ are fresh variables.

We have implemented the algorithm above in Isabelle in 300 lines of SML code. It is available in the Isabelle 2007 distribution.[1] In particular all reflections in this thesis, except §4.3.1, admit automated reification with our method. The reason for §4.3.1 being an exception is that there we have two interpretation functions with the same range type, and hence does not satisfy our first restriction. To deal with these situations, we could extend our algorithm with a completion procedure, but we have not investigated this approach. For example, the previous algorithm returns $(\!|\boldsymbol{v}_0 + \sum_0^5 \widehat{\frac{3}{4}} * \boldsymbol{v}_0{}^{\boldsymbol{2}} + \boldsymbol{v}_1|\!)_\sigma^{[m]}$ when called on the term $\underline{m} + \sum_{k=0}^5 \frac{3}{4} \cdot \underline{k}^2 + m$ and the equations of Figure 2.2.1.

2.3 Certificates

Many interesting statements can be proved by checking a "certificate". It is sufficient, for instance, to provide a truth assignment to prove satisfiability of propositional formulae. We present in the following a method to prove large numbers prime, using Pocklington certificates. A natural number n is defined to be prime if $\mathsf{prime}\ n$ holds, see (2.19). There are many equivalent formulations to primality and (2.20) is useful later.

$$\mathsf{prime}\ n \leftrightarrow p > 1 \wedge \forall m.m \mid p \rightarrow m = 1 \vee m = p \tag{2.19}$$

$$\mathsf{prime}\ n \leftrightarrow n \neq 0 \wedge n \neq 1 \wedge \neg\exists p.\mathsf{prime}\ p \wedge p \mid n \wedge p^2 \leq n \tag{2.20}$$

Two numbers n and m are coprime ($\mathsf{coprime}\ n\ m$) if gcd $n\ m = 1$. For this example we need several number theoretic theorems. Although we proved them all we do not present them here in detail since this shifts the focus of the example. We refer to them by their known names in mathematics, e.g. Bezout identity, Little Fermat or Chinese remainder theorem. Pocklington's theorem in (2.21) is the basis for our method.

$$\begin{aligned} n \geq 2 \wedge n - 1 = q \cdot r \wedge n \leq q^2 \wedge a^{n-1} \equiv 1 (mod\ n) \\ \wedge (\forall p.\mathsf{prime}\ p \wedge p \mid q \rightarrow \mathsf{coprime}\ (a^{\frac{n-1}{p}} - 1)\ n) \rightarrow \mathsf{prime}\ n. \end{aligned} \tag{2.21}$$

Proof of (2.21). First note that from the premises and (2.20) it is sufficient to prove that no prime p exists with $p \mid n \wedge p^2 \leq n$. Assume for contradiction that such a p exists, then it must satisfy $p \leq q$. In the following, we prove $p \equiv 1\ (mod\ q)$, which implies $q \leq p - 1$ and hence a contradiction.

To prove $p \equiv 1\ (mod\ q)$, first note that there are natural number k and l such that $a^{q \cdot r} - 1 = n \cdot k \wedge n = p \cdot l$ and that $a \neq 0$ follows from the premises. Let $\mathsf{ord}\ n\ a$ be the least positive d such that $a^d \equiv 1\ (mod\ n)$, if n and a are coprime and 0 otherwise, for any a and n. Hence $\mathsf{ord}\ p\ a^r \mid q$ and there is a d such that $q = d \cdot \mathsf{ord}\ p\ a^r$. This step entails the application of various forms of the Bezout identity and Euler's generalisation of Fermat's little theorem. Clearly $d \neq 0$ and we show $d = 1$. Assume $d \neq 1$ then it has a prime divisor p', i.e. $\mathsf{prime}\ p' \wedge d = p' \cdot t$ for some t and hence $q = p' \cdot s$ for some s. Hence $p \mid a^{t \cdot r \cdot \mathsf{ord}\ p a^r} - 1$, which contradicts the premises since $\mathsf{coprime}\ (a^{\frac{n-1}{p'}} - 1)\ n$ holds. Now we know that $d = 1$ and hence $\mathsf{ord}\ p\ a^r = q$. Moreover we can easily show that p and a are coprime. Hence $\mathsf{coprime}\ a^r\ p$ and by Fermat's little theorem $p \equiv 1\ (mod\ q)$ □

Primality certificates

To prove $\mathsf{prime}\ n$, according to (2.21), we only need to find r and q such that $q \cdot r = n - 1 \wedge n \leq q^2$, q's prime number decomposition $q_1, \ldots, q_m$, a witness a satisfying $a^{n-1} \equiv 1 (mod\ n)$ and $\mathsf{coprime}(a^{\frac{n-1}{p}} - 1)n$ and certificates $c_1, \ldots, c_m$ for the q_i's that they are indeed prime. Note that we only recur on the primes dividing q, i.e. r can be neglected. We reflect such a certificate by $\mathsf{Decomp}\ q\ r\ a\ [(q_1, k_1, c_1) \ldots, (q_m, k_m, c_m)]$, where k_i represents the multiplicity of q_i, i.e. $\prod_{i=1}^m q_i^{k_i} = q$. Of course there are primes we already know of and for which no

[1] `HOL/ex/Reflection.thy`

further certificate is needed. We know at least that prime 2 holds, a fact which indeed *cannot* be proved using (2.21). This is reflected by the following datatype:

$$\mathsf{datatype}\ cert = \mathsf{Known} \mid \mathsf{Decomp}\ \mathbb{N}\ \mathbb{N}\ \mathbb{N}\ [\mathbb{N} \times \mathbb{N} \times cert]$$

Note that r and a can actually be computed from n and the q_i's, but we include them in the certificate to make checking faster and also easier to verify.

Finding certificates

We search for certificates outside the logical kernel. We make the search dependent on a database of known primes, a table containing for every number n a theorem prime n. At the beginning, the database contains only 2. Now given a candidate n, we first look up if it is in our database, which, if true, makes the proof task trivial. If not, then we decompose $n-1$ into its prime factors and split those so that we find a q satisfying $n \leq q^2$. Note that $r = \frac{n-1}{q}$ and that the prime factors of q are also at hand. The computationally involved task is to find a "primitive root" a satisfying the premise of (2.22). We perform this by incrementally increasing a candidate a and checking if the conditions hold. Note that if n is really prime such a number must exist. For the prime factors of q we proceed recursively as explained. Finding prime number decompositions is not a simple task (it is in FNP, or in P assuming the extended Riemann hypothesis). Here we use an external optimised software [Bel07]. Our approach worked fine for medium sized prime number with up to 100 decimal digits.

Checking certificates

Property (2.22) is a variation on (2.21) better suited for certificate checking. The proof of (2.22) follows by observing that the premises enforce qs to be a factorisation of q into primes. Recall that $(p,k) \in \{\!\{qs\}\!\}$ intuitively means that p is a prime factor of q with multiplicity k. Note the use of foldl instead of foldr, since the former is tail-recursive.

$$\begin{aligned} n \geq 2 \wedge n - 1 = q \cdot r \wedge n \leq q^2 \wedge a^{n-1} \equiv 1 (mod\ n) \wedge \mathsf{foldl}(\lambda r, (p,n).p^n * r)\ 1\ qs = q \\ \wedge (\forall (p,k) \in \{\!\{qs\}\!\}.\mathsf{prime}\ p \wedge \mathsf{coprime}\ (a^{\frac{n-1}{p}-1})\ n) \rightarrow \mathsf{prime}\ n \end{aligned} \tag{2.22}$$

By (2.22) we should be prepared for an efficient execution of $a^m\ mod\ n$ for large m and n. For that we define pmod, below, and prove (2.23).

$$\begin{aligned} &\mathsf{pmod}\ a\ m\ n = \mathbf{if}\ m = 0\ \mathbf{then}\ 1\ \mathbf{else} \\ &\quad \mathbf{let}\ y = \mathsf{pmod}\ x\ \frac{m}{2}\ n\ ;\ z = y^2\ mod\ n\ \mathbf{in}\ \mathbf{if}\ \mathsf{even}\ m\ \mathbf{then}\ z\ \mathbf{else}\ x \cdot z\ mod\ n \end{aligned}$$

$$n \geq 2 \rightarrow \mathsf{pmod}\ a\ m\ n = a^m\ mod\ n \tag{2.23}$$

Now we implement our checker for primality certificates by check, below, and prove (2.24).

$$\begin{aligned} &\mathsf{check}\ ps\ n\ \mathsf{Known} = n \in \{\!\{ps\}\!\} \\ &\mathsf{check}\ ps\ n\ (\mathsf{Decomp}\ q\ r\ a\ qcs) = n \geq 2 \wedge q \cdot r = n - 1 \wedge n \leq q * q \wedge \\ &\quad \mathsf{foldl}(\lambda r, (f,k,c).r \cdot f^k)\ 1\ qcs = q \wedge \\ &\quad \mathbf{let}\ b = \mathsf{pmod}\ a\ r\ n\ \mathbf{in} \\ &\quad \mathsf{pmod}\ b\ q\ n = 1 \wedge \forall (p,k,c) \in \{\!\{qcs\}\!\}.\mathsf{check}\ ps\ p\ c \wedge \mathsf{coprime}\ (\mathsf{pmod}\ b\ \frac{q}{p} n - 1)\ n \end{aligned}$$

$$(\forall p \in \{\!\{ps\}\!\}.\mathsf{prime}\ p) \wedge \mathsf{check}\ ps\ n\ c \rightarrow \mathsf{prime}\ n \tag{2.24}$$

We generate code [BN00, HN07] for check, using arbitrary precision ML integers for natural numbers. Moreover we can safely use the ML `div` and `mod` operations since these behave the same way over $\mathbb{N}$ as in HOL.

Integration

We integrate the method by means of an oracle, which given a number n first finds a Pocklington certificate for n and a list ps of the claimed "known" primes. The oracle then checks that all numbers in ps are indeed in the database and then runs the checker presented above. In case of success, the oracle returns a theorem prime n, and adds it to the database. We proved primality of all primes less than 10^6, some funny ones 1111111111111111111 and 74747474747474747, and some Mersenne numbers $2^n - 1$ for $n \in \{31, 61, 89, 107, 127, 521, 607, 1279, 2281, 3217\}$. Note that the largest one already has 969 decimal digits. The formalisation of this example took 2500 lines of Isabelle proofs and around 200 lines of SML code.

2.4 Comparison

> However, many people have the less nuanced idea that I'm against anything people describe as "reflection", or (presumably if they just read the title of my paper and not the content) that I love all kinds of reflection.
>
> *(J. R. Harrison)*

All presented paradigms have their advantages and drawbacks. I believe that it is not possible to declare one of them as *best*. Especially derived rules and the reflection approaches are so opposed to one another, that nearly every discussion about them ends in philosophical logic, although raised by engineering issues initially. We propose here to compare the three approaches with respect to efficiency, soundness, completeness and logical impacts on the underlying calculus. Before we start, let us classify reflection with respect to its underlying evaluation machinery.

Inference: here reflection becomes an instance of the pure approach. We dismiss this variant in this section.

Reduction: here we use an ML program to reduce (or normalise) arbitrary HOL terms, and accept its result as an equality proof. We call this R-reflection.

ML: here we generate ML code, run it, and accept the result as an equality proof. We refer to this by ML-reflection.

Universality and applicability One main strength of the pure approach is that it has access to the universal data structure of λ-terms. In §2.1 we presented some generic tools, like conversions and tactics, to deal with arbitrary problems. Moreover in ML, we can use efficient data structures such as destructive arrays and references. But even for purely functional data structures and operations, the libraries in ML are much larger and efficient than in HOL. This is of course just a matter of effort, since HOL implementations must be extended by corresponding correctness theorems. We can write functions with complex termination conditions, or even non-terminating functions. All this is not possible in HOL. In particular a semi-decision procedure for first order logic is not reflectable. Of course we can bound the search space by a parameter. The HOL subset of arbitrary first order statements is not reflectable. For that we need a formalisation depending on a variable number of types for the functions and relations occurring in formulae. These can also have a different number of parameters of *different* types. Summed up: the pure approach is universal, reflection is not. Moreover reflection requires a large theorem proving infrastructure. Some reflections we present later in Chapter 4 would be just torture without packages for recursive functions [Sli96, Kra06] and especially the invaluable tailored induction predicates [Sli96, Sli97].

Efficiency and complexity The price of universality is efficiency: λ-terms contain a lot of redundancy, e.g. types. In both of its forms, R- and ML-reflection is more efficient than the pure approach. In our experience, ML-reflection yields a speedup by a factor of 100 over the pure approach. This difference, however, is always a constant factor. It is quite often, however, that many derived rules are slow due to bad programming, e.g. calling the resolution kernel which performs unification etc, where we just need the very basic modus ponens. Some concrete examples demonstrate a speed up by a factor of up to 1000, just by using the right methodology. For the certificate approach, there is a more interesting relation to computational complexity, since checking is much simpler than finding certificates. For instance, recall that we might define NP as the set of all problems having certificates checkable in polynomial time. Hence unless P=NP, the certificate approach is substantially more efficient for *proofs*. Regardless of P vs. NP, some problems which are likely to be in P (e.g. factorisation of an n digits number is in P provided the extended Riemann Hypothesis holds) have succinct certificates (just multiply the factors, which when naively done is quadratic).

Completeness and soundness Derived rules are sound by construction. Reflection, however, requires tolerance and a relaxation in the notion of a formal proof. This issue arises in the evaluation step, whose correctness follows meta-theoretically by "we could have, at least in principle, performed evaluation using rewriting by full inferences". Derived rules might still not cover all the cases. Generally, we need a long debugging process to eliminate lurking problems. Reflection solves this issue in an elegant way: we have a formal proof that the formalised procedure *always* works. Note that in the certificate approach, we have incompleteness of the checker *and* of the external software.

Reflection, meta-theory and the kernel's size Let us consider the logical impact of reflection and the meta-theoretical argument: "we could have, at least in principle" above. Here R- and ML-reflection differ. R-reflection amounts to augment the kernel with a new rule accepting the results of the reduction-machine as equality proofs. The new kernel, although huge, is still LCF and has *constant* size. In ML-reflection, we generate new ML code for every new procedure using the code generator. The resulting kernel has *variable* size and the code generator becomes a rule generator. Although ML-reflection is more efficient, it might be less attractive, on account of this last consequence. Let us consider two further scenarios for reflection. In the first, we have a reflected checker and a very large certificate at hand. So far we instantiated the checker's correctness theorem with the certificate and then run the checker to obtain equality to *True* in order to infer the conclusion. This approach is very inefficient, if the certificate is very large. Since we could have done the instantiation in principle, we can just omit it and run the generated checker (in ML) on an ML certificate. This kind of reflection circumvents instantiation of the correctness theorem. The second scenario raises the problem of omitting explicit instantiations for functions. Assume we have proved a checker *check* correct, i.e. $check\ c\ x \rightarrow P\ x$. It is easy to prove that $check' = \lambda f, x.check(f\ x)\ x$ satisfies $\forall f.check'\ (f\ x)\ x \rightarrow P\ x$. We can generate code for *check'* and implement the parameter f by a certificate generator, e.g. for Pocklington certificates, factorisation of polynomials or even calling a computer algebra system. In principle, we must instantiate f with a HOL-term in order to derive P. In order to omit this instantiation, we should argue that there is a corresponding function definition in HOL, semantically equivalent to the ML function we use. Such arguments, although not trivial, are often done by hand-waving and an intelligent looking smile. Reflection users should very carefully consider all these issues, otherwise, and after enough meta-theory, we can drop formal reasoning altogether.

2.5 Related work

ML was originally introduced as a meta-language layer over LISP in the original LCF system [GMW79] to manipulate terms and theorems, see [GMM+78]. In particular ML was enhanced to *quote* logical entities (mainly terms [MdR94]) in order to tighten the relation to the logic. See [WC07] for an alternative approach of quoting SML within Isar sources, and anti-quoting logical entities within that. Conversions, conversionals, tactics and tacticals are described in [GMW79, Pau87, GM93]. See [Bar00] for optimised rewriting and [Wel95] for using partial evaluation to optimise rewriting in HOL. In [Del02] Coq was augmented by a tactic language, with several user-friendly facilities (e.g. matching the goal against a given pattern), which now became the standard way to write tactics in Coq. Such a language is especially useful in systems like Coq and Isabelle, where most users are not ML programmers. In the special case of Isabelle and Isar, we believe that pursuing the ideas in [WC07] is more interesting: ML programming within the logical context with several means to refer to logical entities.

Computational reflection in theorem proving at least goes back to the meta-functions in [BM81] (now called gold functions [KM97]). In ACL2, reflection is natural since the object language is a subset of LISP, and hence evaluation and reification are almost trivial using LISP's `QUOTE` and `EVAL`. Many researchers have studied reflection [BB02, Har95] and many type-theory based theorem provers, e.g. [TLG06, Jac06], adopted it in their logic, e.g. the ι-rule in the calculus of inductive constructions. The HOL community also used reflection (called pro-forma theorem proving), but always using inferences for evaluation. See [Har95] for a nice survey about reflection in general, not only computational reflection. In Coq a simple generic reification is implemented as `quote`. However, it has far less functionalities than ours: it does not deal with binding and only allows simple environments. Also dealing with atoms is much more restrictive.

The first serious applications of the certificate approach (called skeptical approach) in theorem proving were presented in [HT98]. The skeptical combination of computer algebra systems and theorem provers has also been studied in [Bal99]. See [Obu05] for proving bounds on linear programs using certificates (Farkas's lemma). See also [KW07] for an interesting alternative of implementing computer algebra facilities on top of HOL Light. An interesting alternative approach to combine computation and deduction also appears in [Far07]. Using Pocklington's theorem and an efficient library [GT06] for modular arithmetic to prove large numbers prime, has been presented in [GTW06]. Our method in §2.3 proves all numbers prime they also prove. For more efficient methods using elliptic curves, see [TH07].

The problem of programming errors in decision procedures has recently been addressed using dependent types [AF04, KS04]. But it seems unlikely that the procedures we present in this thesis can be dealt with automatically in such a framework. This approach does not guarantee completeness. Indeed, missing cases and proofs that fail at run-time are not detected.

Chapter 3

Certificates for polynomial problems

Contents

The goal of this chapter is to present proof methods for simple problems involving polynomials, based on certificates. First, we present formalisations of univariate and multivariate polynomials in §3.1. In §3.2 we use extended Gröbner bases to prove universal statements and an interesting subset of the $\forall\exists$ fragment in rings. In §3.3 we use semidefinite programming to prove universal inequalities of real polynomials.

Notation Let R be a ring, then $R[x]$ denotes the ring of univariate polynomials with coefficients in R. Similarly $R[x_1, \ldots, x_n]$ denotes the ring of multivariate polynomials over R in n variables. We also write $R[\vec{x}]$ for $R[x_1, \ldots, x_n]$. We denote the degree of $p \in R[x]$ by ∂p. A non-empty $I \subseteq R$ is an ideal of R if $\forall x, y \in I.x + y \in I$ and $\forall r \in R.\forall x \in I.r \cdot x \in I$. For $a_1, \ldots, a_n \in R$ the set $\langle a_1, \ldots, a_n\rangle = \{\sum_{i=1}^n a_i \cdot x_i \mid x_1, \ldots, x_n \in R\}$ is an ideal of R and is generated by $a_1, \ldots, a_n$. Let I be an ideal of R, then $\sqrt{I} = \{x|\exists n.x^n \in I\}$ is the radical ideal of I. Note that $I \subseteq \sqrt{I}$. Let $\{p_1, \ldots, p_n\} \subseteq R$ and $q \in R$, then we call $r_1, \ldots, r_n$ a certificate for $q \in \langle p_1, \ldots, p_n\rangle$ if $q = \sum_{i=1}^n r_i \cdot p_i$ holds.

3.1 Polynomials

The focus of our formalisations of polynomials is to be able to compute and prove with them. The formalisations we present have a strong algorithmic flavour. In §3.1.1 we formalise univariate polynomials as "functions": given a list of coefficients $c_0, \ldots, c_n$, they describe a function $x \mapsto \sum_{i=0}^n c_i \cdot x^i$. This approach was successfully used quite early [HSA06, Har06], but due to the lack of classes or equivalent specification mechanisms, the formalisations are duplicated for $\mathbb{R}$ and $\mathbb{C}$. Our formalisation is done in a **locale** context, and hence carries over to several instances including $\mathbb{Z}, \mathbb{R}$ and $\mathbb{C}$. In §3.1.2, we present a formalisation of multivariate polynomials (also in **locale**).

3.1.1 A formalisation of univariate polynomials as functions

We formalise *univariate* polynomials as functions. Given a list of coefficients $[c_0, \ldots, c_n]$, then $\overline{[c_0, \ldots, c_n]} = \lambda x. \sum_{i=0}^n c_i \cdot x^i$.

$$\textbf{fun (in } \mathit{semiring}_0\textbf{)} : \ \overline{[]}\ x = 0 \qquad \mid \overline{c \cdot cs}\ x = (\overline{cs}\ x) \cdot x + c \tag{3.1}$$

This formalisation is very appealing to switch views: a) view polynomials syntactically as a list of coefficients and b) view a list of coefficients p as a polynomial *function* $\overline{p}$. The zero polynomial is $\overline{[]}$. Note that the list representation is not unique, since we can always append zeros and $\overline{p} = \overline{p@[0]}$.

The first step is to implement algorithms for the usual operations on the syntax and prove them correct. The definitions of these algorithms take place in locales with minimal assumptions (mostly $semiring_0$, i.e. a semi-ring with addition and neutral element 0). The formalisation includes addition $\overline{+}$, multiplication $\overline{\cdot}$, exponentiation $\lambda p, n.p^n$, subtraction and negation both denoted by $\overline{-}$, degree $\mathsf{deg}\ p$, normalisation to remove superfluous zero coefficients, the multiplicity $\mathsf{m}_p(a)$ of a root a in p and other utilities we do not present like square free conditions. The definitions of addition and multiplication etc over the syntax are straightforward. Here is for instance the definition of addition:

$$\textbf{fun (in } semiring_0\textbf{)}: \ [] \,\overline{+}\, q = q \quad | \ p \,\overline{+}\, [] = p \quad | \ (c \cdot cs) \,\overline{+}\, (d \cdot ds) = (c + d) \cdot (cs \,\overline{+}\, ds) \tag{3.2}$$

We prove all these operations correct in classes with as few assumptions as possible (e.g. for $\overline{\cdot}$ we need commutativity, cf. (3.3)).

$$\textbf{(in } commsemiring_0\textbf{)}\ \overline{p \,\overline{\cdot}\, q}\ x = (\overline{p}\ x) \cdot (\overline{q}\ x) \tag{3.3}$$

We present here only a few interesting theorems, just to give a rough idea about the formalisation. Like in algebra texts, stronger properties hold in classes with more axioms. The key property (3.4) for roots-factorisation holds e.g. in commutative rings with unity:

$$\textbf{(in } commring_1\textbf{)}: \ \overline{p}\ a = 0 \leftrightarrow p = [] \vee \exists q.p = [-a, 1] \,\overline{\cdot}\, q \tag{3.4}$$

Moreover in an integral domain of characteristic zero ($idom_0$), every polynomial has finitely many roots exactly when it is not the zero polynomial (cf. (3.5)) and the entirety property (3.6) holds. Here another strong and interesting property holds: a polynomial (*as a function*) is zero exactly when all its coefficients are zero (cf. (3.7)). This last property plays a central role in the implementation and also for the uniqueness property of multivariate polynomials in §3.1.2.

$$\textbf{(in } idom_0\textbf{)}: \ \overline{p} \neq \overline{[]} \leftrightarrow \mathsf{finite}\{x | \overline{p}\ x = 0\} \tag{3.5}$$

$$\textbf{(in } idom_0\textbf{)}: \ \overline{p \,\overline{\cdot}\, q} = \overline{[]} \leftrightarrow \overline{p} = \overline{[]} \vee \overline{q} = \overline{[]} \tag{3.6}$$

$$\textbf{(in } idom_0\textbf{)}: \ \overline{p} = \overline{[]} \leftrightarrow \forall c \in \{\!\{p\}\!\}.c = 0 \tag{3.7}$$

We also formalise the multiplicity $\mathsf{m}_p(a)$ of a polynomial p's root a, the degree of polynomials ($\mathsf{deg}\ p$) and square free decompositions. In §4.4 we shall use divisibility ($|$), cf. (3.8). Let p^n denote the n^{th} power of p (iterative application of $\overline{\cdot}$) and $p \nmid q$ denote $\neg p \mid q$. Any non-zero polynomial $p(x)$ can be decomposed into a product $(x - a)^m \cdot q(x)$, where m is a's multiplicity in p and a is not a zero of q, see (3.9).

$$\textbf{fun (in } semiring_0\textbf{)}: \ p \mid q \leftrightarrow \exists r.\overline{q} = \overline{p \,\overline{\cdot}\, r} \tag{3.8}$$

$$\textbf{(in } idom_0\textbf{)}: \ \overline{p} \neq \overline{[]} \rightarrow \exists q.\overline{p} = \overline{[-a, 1]^{\mathsf{m}_p(a)} \,\overline{\cdot}\, q} \wedge [-a, 1] \nmid q \tag{3.9}$$

The formalisation in (3.1) is very suitable for abstract reasoning about *univariate* polynomials. We can also generate code if the underlying (semi)ring allows it. Note the ability of the code-generator framework [HN07] to deal with classes using dictionaries. Note that the generated code depends on the implementation of the operations and hence *cannot* be used to compute abstractly. The main drawback is that it does not "naturally" carry over to multivariate polynomials. The type corresponding to $R[x_1, \ldots, x_n]$ would be $[\ldots[\alpha]\ldots]$, nested n times, which can not be expressed in HOL. This formalisation is 1000 lines long.

3.1.2 Reflected multivariate polynomial utilities

We present here an executable formalisation of *multivariate* polynomials over rings. As in §3.1.1 stronger theorems only hold in classes with more axioms. To generate code running for all instances, we restrict the coefficients to the sub-ring $\{\underline{i} \mid i \in \mathbb{Z}\}$, i.e. constants $\underline{i}$ injected from $i \in \mathbb{Z}$. "Injected" does not mean that $\lambda i.\underline{i}$ is injective, this is true only for rings of characteristic zero.

$$\mathsf{datatype}\ \boldsymbol{\rho} \quad = \widehat{\mathbb{Z}}|\boldsymbol{v}_{\mathbb{N}}| - \boldsymbol{\rho}|\boldsymbol{\rho} + \boldsymbol{\rho}|\boldsymbol{\rho} - \boldsymbol{\rho}|\boldsymbol{\rho} * \boldsymbol{\rho}|\boldsymbol{\rho}^{\mathbb{N}}$$

$$\begin{array}{ll} (\!|\widehat{c}|\!)^e_{\boldsymbol{\rho}} & = \underline{c} \\ (\!|\boldsymbol{v}_n|\!)^e_{\boldsymbol{\rho}} & = e!n \\ (\!|-\ p|\!)^e_{\boldsymbol{\rho}} & = -(\!|p|\!)^e_{\boldsymbol{\rho}} \end{array} \qquad \begin{array}{ll} (\!|p + q|\!)^e_{\boldsymbol{\rho}} & = (\!|p|\!)^e_{\boldsymbol{\rho}} + (\!|q|\!)^e_{\boldsymbol{\rho}} \\ (\!|p - q|\!)^e_{\boldsymbol{\rho}} & = (\!|p|\!)^e_{\boldsymbol{\rho}} - (\!|q|\!)^e_{\boldsymbol{\rho}} \\ (\!|p * q|\!)^e_{\boldsymbol{\rho}} & = (\!|p|\!)^e_{\boldsymbol{\rho}} \cdot (\!|q|\!)^e_{\boldsymbol{\rho}} \\ (\!|p^{\boldsymbol{n}}|\!)^e_{\boldsymbol{\rho}} & = ((\!|p|\!)^e_{\boldsymbol{\rho}})^n \end{array}$$

Figure 3.1: Syntax and semantics of polynomial expressions

The syntax (datatype $\boldsymbol{\rho}$) and its semantics in Figure 3.1 reflect multivariate polynomial expressions. The semantics $(\!|.|\!)_{\boldsymbol{\rho}}$ is parametrised by an environment e (a list of "ring-elements"). We represent variables by de Bruijn indices: $\boldsymbol{v}_n$ represents the bound variable with index $n :: \mathbb{N}$. Note that $(\!|\boldsymbol{v}_n|\!)^e_{\boldsymbol{\rho}} = e!n$ is the n^{th} element of e. The bold symbols $+, *$ etc are constructors and reflect their counterparts $+, \cdot$ etc in the logic. We reflect $\underline{i}$ by $\widehat{i}$, i.e.$\widehat{\ }$is a constructor of $\boldsymbol{\rho}$.

The normal form defined by ishorn imposes a Horner scheme for multivariate polynomials where the ordering on variables is induced by their indices.

$$\begin{array}{ll} \mathsf{ishornh}\ \widehat{c}\ n & = \mathit{True} \\ \mathsf{ishornh}\ (c + \boldsymbol{v}_m * p)\ n & = p \neq \boldsymbol{0}_{\boldsymbol{\rho}} \wedge m \geq n \wedge \mathsf{ishornh}\ c\ (m+1) \wedge \mathsf{ishornh}\ p\ m \\ \mathsf{ishorn}\ p & = \mathsf{ishornh}\ p\ 0 \end{array}$$

For example $5 \cdot y \cdot x^2 + y \cdot (x+2)$ is reflected by $(\!|q|\!)^{[x,y]}_{\boldsymbol{\rho}} = (\!|r|\!)^{[x,y]}_{\boldsymbol{\rho}}$, for $q = \widehat{5} * \boldsymbol{v}_1 * \boldsymbol{v}_0{}^{\boldsymbol{2}} + \boldsymbol{v}_1 * (\boldsymbol{v}_0 + \widehat{2})$ and $r = (\widehat{0} + \boldsymbol{v}_1 * \widehat{2}) + \boldsymbol{v}_0 * ((\widehat{0} + \boldsymbol{v}_1 * \widehat{1}) + \boldsymbol{v}_0 * (\widehat{0} + \boldsymbol{v}_1 * \widehat{5}))$. Only r is in normal form.

Ultimately we would like to compute the normal form of arbitrary $\boldsymbol{\rho}$-polynomials. We show subsequently that it is legitimate to write *the* normal form, since we show uniqueness in a certain sense: two $\boldsymbol{\rho}$-polynomials in normal form are *syntactically* equal if and only if their interpretations are equal in *all* possible environments. We present a function $\mathsf{norm}_{\boldsymbol{\rho}}$ to normalise any $\boldsymbol{\rho}$-polynomial. For this, it applies algorithms for addition ($\oplus$), multiplication ($\circledast$), negation and subtraction ($\ominus$) and power $\lambda p, n.p^{\downarrow \boldsymbol{n}}$, with the additional property that they preserve the normal form. See Figure 3.2 for the definitions of addition, multiplication and power. Subtraction and negation are straightforward. Taking p to the power of n repeatedly applies $\circledast$ depending on the binary scheme of n. All these operations preserve normal form, cf. (3.10), and semantics, cf. (3.11 – 3.13). Now the definition of $\mathsf{norm}_{\boldsymbol{\rho}}$ is simple and mainly replaces the constructors with the definitions above. The main property of $\mathsf{norm}_{\boldsymbol{\rho}}$ in (3.14) uses (3.10 – 3.13) is proved by structural induction. For $\ominus$ there is a syntactical property, cf. (3.15).

$$\mathsf{ishorn}\ p \rightarrow \mathsf{ishorn}\ p^{\downarrow \boldsymbol{n}} \wedge (\mathsf{ishorn}\ q \rightarrow \mathsf{ishorn}\ (p \oplus q) \wedge \mathsf{ishorn}\ (p \circledast q)) \tag{3.10}$$

$$\textbf{(in } \mathit{ring}_1\textbf{)}: \ (\!|p \oplus q|\!)^e_{\boldsymbol{\rho}} = (\!|p + q|\!)^e_{\boldsymbol{\rho}} \tag{3.11}$$

$$\textbf{(in } \mathit{commring}_1\textbf{)}: \ (\!|p \circledast q|\!)^e_{\boldsymbol{\rho}} = (\!|p * q|\!)^e_{\boldsymbol{\rho}} \tag{3.12}$$

$$\textbf{(in } \mathit{commring}_1\textbf{)}: \ (\!|p^{\downarrow \boldsymbol{n}}|\!)^e_{\boldsymbol{\rho}} = (\!|p^{\boldsymbol{n}}|\!)^e_{\boldsymbol{\rho}} \tag{3.13}$$

$$\textbf{(in } \mathit{commring}_1\textbf{)}: \ \mathsf{ishorn}\ (\mathsf{norm}_{\boldsymbol{\rho}}\ p) \wedge (\!|\mathsf{norm}_{\boldsymbol{\rho}}\ p|\!)^e_{\boldsymbol{\rho}} = (\!|p|\!)^e_{\boldsymbol{\rho}} \tag{3.14}$$

$$\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \rightarrow (p \ominus q = \boldsymbol{0}_{\boldsymbol{\rho}}) \leftrightarrow (p = q) \tag{3.15}$$

Using (3.14) we obtain an incomplete method to prove equality of two polynomials and hence cover all the results of [GM05]. We show completeness later.

Miscellaneous utilities and syntactical properties

We have a few further utilities mainly motivated by the algorithms in §4.4 and §4.5, which mostly view a multivariate polynomial as univariate in $\boldsymbol{v}_0$. Figure 3.3 exhibits some of them:

$$
\begin{array}{lcl}
\widehat{c} \oplus \widehat{d} & = & \widehat{c+d} \\
\widehat{c} \oplus (d + \boldsymbol{v}_k * q) & = & (\widehat{c} \oplus d) + \boldsymbol{v}_k * q \\
(c + \boldsymbol{v}_n * p) \oplus \widehat{d} & = & (c \oplus \widehat{d}) + \boldsymbol{v}_n * p \\
(c + \boldsymbol{v}_n * p) \oplus (d + \boldsymbol{v}_k * q) & = & \textbf{if } n < k \textbf{ then } (c \oplus (d + \boldsymbol{v}_k * q)) + \boldsymbol{v}_n * p \\
 & & \textbf{else if } k < n \textbf{ then } ((c + \boldsymbol{v}_n * p) \oplus d) + \boldsymbol{v}_k * q \\
 & & \textbf{else let } cd = c \oplus d; pq = p \oplus q \\
 & & \quad \textbf{in if } pq = \mathbf{0}_{\boldsymbol{\rho}} \textbf{ then } cd \textbf{ else } cd + \boldsymbol{v}_n * pq \\
a \oplus b & = & a + b \\
 & & \\
\widehat{c} \circledast \widehat{d} & = & \widehat{c \cdot d} \\
\widehat{c} \circledast (d + \boldsymbol{v}_k * q) & = & \textbf{if } c = 0 \textbf{ then } \mathbf{0}_{\boldsymbol{\rho}} \textbf{ else } (\widehat{c} \circledast d) + \boldsymbol{v}_k * (\widehat{c} \circledast q) \\
(c + \boldsymbol{v}_n * p) \circledast \widehat{d} & = & \textbf{if } d = 0 \textbf{ then } \mathbf{0}_{\boldsymbol{\rho}} \textbf{ else } (\widehat{d} \circledast c) + \boldsymbol{v}_n * (\widehat{d} \circledast p) \\
(c + \boldsymbol{v}_n * p) \circledast (d + \boldsymbol{v}_k * q) & = & \textbf{if } n < k \textbf{ then} \\
 & & \quad (c \circledast (d + \boldsymbol{v}_k * q)) + \boldsymbol{v}_n * (p \circledast (d + \boldsymbol{v}_k * q)) \\
 & & \textbf{else if } k < n \textbf{ then} \\
 & & \quad ((c + \boldsymbol{v}_n * p) \circledast d) + \boldsymbol{v}_k * ((c + \boldsymbol{v}_n * p) \circledast q) \\
 & & \textbf{else } ((c + \boldsymbol{v}_n * p) \circledast d) \oplus (\mathbf{0}_{\boldsymbol{\rho}} + \boldsymbol{v}_n * ((c + \boldsymbol{v}_n * p) \circledast q)) \\
a \circledast b & = & a * b \\
 & & \\
p^{\downarrow \mathbf{0}} & = & \mathbf{1}_{\boldsymbol{\rho}} \\
p^{\downarrow \boldsymbol{n}} & = & \textbf{let } q = p^{\downarrow \frac{\boldsymbol{n}}{\mathbf{2}}}; d = q \circledast q \\
 & & \textbf{in if } \mathsf{even}\ n \textbf{ if } d \textbf{ else } p \circledast d
\end{array}
$$

Figure 3.2: Addition, multiplication and power for $\boldsymbol{\rho}$-polynomials

$$
\begin{array}{ll}
\mathsf{coeffs}\ (c + \boldsymbol{v}_0 * p) & = c \cdot \mathsf{coeffs}\ p \\
\mathsf{coeffs}\ p & = [p] \\
\mathsf{deg}_{\boldsymbol{\rho}}\ m\ (c + \boldsymbol{v}_n * p) & = \textbf{if } n = m \textbf{ then } 1 + \mathsf{deg}_{\boldsymbol{\rho}}\ m\ p \textbf{ else } 0 \\
\mathsf{deg}_{\boldsymbol{\rho}}\ m\ p & = 0 \\
\mathsf{head}\ m\ (c + \boldsymbol{v}_n * p) & = \textbf{if } n = m \textbf{ then } \mathsf{head}\ m\ p \textbf{ else } c + \boldsymbol{v}_n * p \\
\mathsf{head}\ m\ p & = p
\end{array}
$$

Figure 3.3: Degree, head and coefficients of $\boldsymbol{\rho}$-polynomials

function coeffs returns the list of coefficients of a normalised polynomial seen as univariate in $\boldsymbol{v}_0$ (this is a list of multivariate polynomials in the remaining variables) and functions $\mathsf{deg}_{\boldsymbol{\rho}}$ and head return the degree and head (resp.) of a normalised polynomial seen as univariate in $\boldsymbol{v}_n$, *where no* $\boldsymbol{v}_m$ *occurs with* $m < n$. These correspond exactly to the notion of head and degree for $m = 0$. Note that the head of a polynomial is its last coefficient and its degree is one less than the length of the coefficients:

$$\mathsf{ishorn}\ p \rightarrow \mathsf{last}(\mathsf{coeffs}\ p) = \mathsf{head}\ 0\ p \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p = |\mathsf{coeffs}\ p| - 1. \tag{3.16}$$

To start with, the theorems about $\oplus$ and $\mathsf{deg}_{\boldsymbol{\rho}}$ are simple to prove. Adding two polynomials yields a polynomial with degree less or equal than their maximal degree (3.18) and equality is certainly achieved for polynomials with different degrees (3.17). Finally (3.19) states that if two polynomials add to a constant, then they must have the same degree. Subtracting two polynomials with the same head and degree must yield a polynomial with a *strictly* smaller degree or $\mathbf{0}_{\boldsymbol{\rho}}$, cf. (3.20). Property (3.20) plays an important role in the

correctness and termination proof of pseudo-division.

$$\begin{aligned}
&\mathsf{ishornh}\ p\ n_0 \wedge \mathsf{ishornh}\ q\ n_1 \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ m\ p \neq \mathsf{deg}_{\boldsymbol{\rho}}\ m\ q \wedge m \leq \min\ n_0\ n_1 \\
&\quad \rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ m\ (p \oplus q) = \max\ (\mathsf{deg}_{\boldsymbol{\rho}}\ m\ p\ m)\ (\mathsf{deg}_{\boldsymbol{\rho}}\ m\ q) && (3.17)\\
&\mathsf{ishornh}\ p\ n_0 \wedge \mathsf{ishornh}\ q\ n_1 \wedge m \leq \max\ n_0\ n_1 \\
&\quad \rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ m\ (p \oplus\ q)\ \leq \max\ (\mathsf{deg}_{\boldsymbol{\rho}}\ m\ p)\ (\mathsf{deg}_{\boldsymbol{\rho}}\ m\ q) && (3.18)\\
&\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \wedge p \oplus q = \widehat{c} \rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ m\ p = \mathsf{deg}_{\boldsymbol{\rho}}\ m\ q && (3.19)\\
&\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \wedge \mathsf{head}\ 0\ p = \mathsf{head}\ 0\ q \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p = \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ q \\
&\quad \rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ (p \ominus q) < \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p \vee p \ominus q = \mathbf{0}_{\boldsymbol{\rho}} && (3.20)
\end{aligned}$$

The following properties connecting $\mathsf{deg}_{\boldsymbol{\rho}}$ and $\circledast$ have to be proved simultaneously altogether by induction on p and q:

$$\begin{aligned}
&\mathsf{ishornh}\ p\ n_0 \wedge \mathsf{ishornh}\ q\ n_1 \wedge m \leq \min\ n_0\ n_1 \\
&\rightarrow \mathsf{ishornh}\ (p \circledast q)\ (\min\ n_0\ n_1) \wedge p \circledast q = \mathbf{0}_{\boldsymbol{\rho}} \leftrightarrow p = \mathbf{0}_{\boldsymbol{\rho}} \vee q = \mathbf{0}_{\boldsymbol{\rho}} \\
&\quad \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ m\ (p \circledast q) = \mathbf{if}\ p = \mathbf{0}_{\boldsymbol{\rho}} \vee q = \mathbf{0}_{\boldsymbol{\rho}}\ \mathbf{then}\ 0 \\
&\qquad\qquad \mathbf{else}\ \mathsf{deg}_{\boldsymbol{\rho}}\ m\ p + \mathsf{deg}_{\boldsymbol{\rho}}\ m\ q
\end{aligned} \tag{3.21}$$

Finally the main theorems about head hold as expected:

$$\begin{aligned}
&\mathsf{ishornh}\ p\ n_0 \wedge \mathsf{ishornh}\ q\ n_1 \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p \neq \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ q \\
&\rightarrow \mathsf{head}\ 0\ (p \oplus q) = \ \mathbf{if}\ \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p < \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ q\ \mathbf{then}\ \mathsf{head}\ 0\ q\ \mathbf{else}\ \mathsf{head}\ 0\ p && (3.22)\\
&\mathsf{ishornh}\ p\ n_0 \wedge \mathsf{ishornh}\ q\ n_1\ \ \wedge p \neq \mathbf{0}_{\boldsymbol{\rho}} \wedge q \neq \mathbf{0}_{\boldsymbol{\rho}} \\
&\quad \rightarrow \mathsf{head}\ 0\ (p \circledast q) = \mathsf{head}\ 0\ p \circledast \mathsf{head}\ 0\ q. && (3.23)
\end{aligned}$$

We also formalise several other notions such as the fact that $\boldsymbol{v}_0$ does not occur in a polynomial p (by $\mathsf{unbound}_{\boldsymbol{\rho}}\ p$) and decrementing all de Bruijn indices (by $\mathsf{decr}_{\boldsymbol{\rho}}\ p$). The following simple property holds.

$$(\mathbf{in}\ \mathit{ring}_1)\ \mathsf{unbound}_{\boldsymbol{\rho}}\ p \rightarrow (\!|\mathsf{decr}_{\boldsymbol{\rho}}\ p|\!)^e_{\boldsymbol{\rho}} \leftrightarrow (\!|p|\!)^{x \cdot e}_{\boldsymbol{\rho}} \tag{3.24}$$

Multivariate vs. univariate polynomials and the uniqueness property

So far we considered multivariate polynomials as univariate implicitly: the implementation treats $\boldsymbol{v}_0$ in a special manner. Given an environment e, then function $\lambda p.[p]^e$ in (3.25) connects $\boldsymbol{\rho}$-polynomials to the univariate ones of §3.1.1. Obviously it satisfies (3.26). This simple connection transfers properties from §3.1.1 to $\boldsymbol{\rho}$-polynomials and is invaluable in §4.5.

$$(\mathbf{fun}\ (\mathbf{in}\ \mathit{ring}_1) :\ [p]^e = \mathsf{map}\ (\lambda q.(\!|\mathsf{decr}_{\boldsymbol{\rho}}\ q|\!)^e_{\boldsymbol{\rho}})\ (\mathsf{coeffs}\ p) \tag{3.25}$$

$$(\mathbf{in}\ \mathit{ring}_1) :\ \mathsf{ishornh}\ p\ n_0 \rightarrow (\!|p|\!)^{x \cdot e}_{\boldsymbol{\rho}} = \overline{[p]^e}\ x \tag{3.26}$$

One of the strongest and most desirable properties is the uniqueness of normalised $\boldsymbol{\rho}$-polynomials, since it expresses a tight connection between syntax and semantics. This is almost analogous to (3.7) in §3.1.1, which is central in the uniqueness proof.

Let $\mathsf{max}^{\boldsymbol{v}}_p$ denote the maximal n, such that $\boldsymbol{v}_n$ occurs in p. Property (3.27) is the analogue of (3.7) and states that a normalised polynomial is $\mathbf{0}_{\boldsymbol{\rho}}$ exactly when it evaluates to 0 for any reasonable environment e. The proof of (3.27) is by complete induction on $\mathsf{max}^{\boldsymbol{v}}_p$ where (3.7) is applied to the coefficients of $[p]^e$. This corresponds to an induction over the number of variables n of the multivariate polynomial ring $R[x_1, \ldots, x_n]$. From (3.15) and (3.27) we derive the uniqueness property (3.28).

$$(\mathbf{in}\ \mathit{idom}_0) :\ \mathsf{ishorn}\ p \rightarrow (\forall e.|e| \geq \mathsf{max}^{\boldsymbol{v}}_p \rightarrow (\!|p|\!)^e_{\boldsymbol{\rho}} = 0) \leftrightarrow p = \mathbf{0}_{\boldsymbol{\rho}} \tag{3.27}$$

$$(\mathbf{in}\ \mathit{idom}_0) :\ \mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \rightarrow (\forall e.(\!|p|\!)^e_{\boldsymbol{\rho}} = (\!|q|\!)^e_{\boldsymbol{\rho}}) \leftrightarrow p = q \tag{3.28}$$

An important impact of (3.28) is that all interesting algebraic properties about $+, \cdot$, etc. ultimately carry over to $\oplus, \circledast$, etc *on the syntax-level*, e.g. distributivity and commutativity hold for $\oplus$ and $\circledast$. The only drawback we must accept is that the properties proved in this manner, although purely syntactical, hold only inside idom_0.

$$\begin{array}{lcl}
\mathsf{pdiv}\ a\ n\ p\ k\ s & = & \textbf{if } s = \mathbf{0}_{\boldsymbol{\rho}} \textbf{ then } (k, s) \textbf{ else} \\
& & \textbf{let } b = \mathsf{head}\ 0\ s; m = \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ s \textbf{ in} \\
& & \textbf{if } m < n \textbf{ then } (k, s) \textbf{ else} \\
& & \textbf{let } p' = (\mathbf{0}_{\boldsymbol{\rho}} + \boldsymbol{v}_0 * \mathbf{1}_{\boldsymbol{\rho}})^{\downarrow \boldsymbol{m-n}} \circledast p \textbf{ in} \\
& & \textbf{if } a = b \textbf{ then } \mathsf{pdiv}\ a\ n\ p\ k\ (s \ominus p)\ p' \\
& & \textbf{else } \mathsf{pdiv}\ a\ n\ p\ (k+1)\ (a \circledast s \ominus b \circledast p') \\
s \oslash p & = & \mathsf{pdiv}\ (\mathsf{head}\ 0\ p)\ (\mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p)\ p\ 0\ s
\end{array}$$

Figure 3.4: Pseudo-division of $\boldsymbol{\rho}$-polynomials

Pseudo-division

Pseudo-division is a central algorithm in many procedures we consider (§4.4 and §4.5). Pseudo-dividing $s(x)$ by $p(x)$ yields a quotient $q(x)$ and remainder $r(x)$ and a constant c, i.e. a polynomial not involving x, such that

$$c \cdot s(x) = p(x) \cdot q(x) + r(x) \wedge \partial r < \partial p$$

For *univariate* polynomials with rational coefficients, we can ensure $c = 1$, but in the multivariate case we can only ensure $c = a^k$, where a is p's head, i.e.

$$a^k \cdot s(x) = p(x) \cdot q(x) + r(x) \wedge \partial r < \partial p$$

Pseudo-division of s by p asks for k, q and r above. Suppose $p(x) = a \cdot x^n + p_0(x)$ and $s(x) = b \cdot x^m + s_0(x)$, where $\partial p_0 < \partial p$ and $\partial s_0 < \partial s$. If $m < n$ then $q(x) = 0, r(x) = s(x)$ and $k = 0$. For $m \geq n$, first note that $a \cdot s(x) = b \cdot x^{m-n} \cdot p(x) + s'(x)$, where $s'(x) = a \cdot s_0(x) - b \cdot x^{m-n} \cdot p_0(x)$. Note that $\partial s' < \partial s$, since the heads cancel each other. Assume now that recursively $a^{k'} \cdot s'(x) = q'(x) \cdot p(x) + r'(x)$ holds, then

$$\begin{aligned}
a^{k'+1} \cdot s(x) &= a^{k'} \cdot b \cdot x^{m-n} \cdot p(x) + a^{k'} \cdot s'(x) \\
&= (a^{k'} \cdot b \cdot x^{m-n} + q'(x)) \cdot p(x) + r'(x)
\end{aligned}$$

and hence $k = k' + 1$, $q(x) = a^{k'} \cdot b \cdot x^{m-n} + q'(x)$ and $r(x) = r'(x)$ is a correct answer. Note also that if $a = b$ it is not necessary to multiply by a (i.e. increase k).

Given two $\boldsymbol{\rho}$-polynomials s and p, $s \oslash p$ in Figure 3.4 returns r and k as above (in our applications q is not relevant). The main recursion happens in pdiv, where the parameters are assumed to satisfy (D) $\mathsf{head}\ 0\ p = a \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p = n \wedge p \neq \mathbf{0}_{\boldsymbol{\rho}} \wedge \mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ s$.

We use Isabelle's facility for partial functions [Kra06] to define pdiv. Note that termination is *not* guaranteed for non-normalised polynomials. As a consequence pdiv has no induction principle but a *partial* one allowing to apply the induction hypothesis only after proving the recursive calls to be in pdiv's domain. It is hence necessary to prove that pdiv's domain at least contains the parameters satisfying (D), cf. (3.29). The main property of pdiv is in (3.30). Note the syntactical nature of the pseudo-division property.

$$\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ s \wedge \mathsf{head}\ 0\ p = a \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p = n \wedge p \neq \mathbf{0}_{\boldsymbol{\rho}} \rightarrow (a, n, p, k, s) \in \mathsf{dom}(\mathsf{pdiv}) \tag{3.29}$$

$$\begin{aligned}
\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ s \wedge \mathsf{head}\ 0\ p = a \wedge \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p = n \wedge p \neq \mathbf{0}_{\boldsymbol{\rho}} \wedge \mathsf{pdiv}\ a\ n\ p\ k\ s = (k', r) \\
\rightarrow k' \geq k \wedge \mathsf{ishorn}\ r \wedge (\mathsf{deg}_{\boldsymbol{\rho}}\ 0\ r = 0 \vee \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ r < \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p) \\
\wedge (\exists q. \mathsf{ishorn}\ q \wedge a^{\downarrow \boldsymbol{k'-k}} \circledast s = p \circledast q \oplus r))
\end{aligned} \tag{3.30}$$

The proof of (3.30) is by complete induction on $\mathsf{deg}_{\boldsymbol{\rho}}\ 0\ s$. The decomposition proof follows the sketch above, whereas the entailment into the domain of recursive calls massively uses the syntactical properties and the uniqueness presented before. Defining pseudo-division and proving (3.30) is a challenge for any formalised library of polynomial utilities. This formalisation is 1700 lines long. In §3.2 and §3.3 we use our formalisations in reflected proof methods.

3.2 Equations and disequations

In this section we consider two classes of formulae involving equations ($p = 0$) and disequations[1] ($q \neq 0$), for polynomials p and q: a) universal formulae and b) a subset of $\forall\exists$ statements, which covers some interesting number theoretic properties. The formulae in b) have the form

$$\forall \vec{x}. \bigwedge_i e_i(\vec{x}) = 0 \rightarrow \exists \vec{y}. \bigwedge_{i=1}^{k} \sum_{j=1}^{m} p_{ij}(\vec{x}) \cdot y_j = a_i(\vec{x}),$$

where $\vec{x}$ and $\vec{y}$ are shorthand for $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ resp., $e_i(\vec{x}), a_i(\vec{x})$ and $p_{ij}(\vec{x})$ are polynomials in $x_1, \ldots, x_n$ for every $1 \leq i \leq n$ and $1 \leq j \leq m$. The main tool both methods use is the extended Gröbner bases algorithm (see [BWK93] or [Mis93]) to find a certificate $q_1, \ldots, q_n$ for $p \in \langle p_1, \ldots, p_n \rangle$, i.e. $\sum_{i=1}^{n} q_i \cdot p_i = p$ holds.

3.2.1 The universal case

Consider a universal formula P with equations ($p = 0$) and disequations ($q \neq 0$) as atoms, where p and q are multivariate polynomials over an integral domain. We prove P by refuting $\neg P$ and the latter by refuting each disjunct $\bigwedge_{i=1}^{n} p_i = 0 \wedge \bigwedge_{j=1}^{m} q_j \neq 0$ of its DNF. We might safely assume that $m = 1$ (if $m = 0$ just add $1 \neq 0$, otherwise $\bigwedge_{j=1}^{m} q_j \neq 0 \leftrightarrow q \neq 0$, where $q = \prod_{j=1}^{m} q_i$). If $n = 0$ then normalise q and compare it to 0. Otherwise, assume $n > 0$. Clearly proving $\sum_{i=1}^{n} f_i \cdot p_i = q^k$ for some k and coefficients f_i (i.e. $q \in \sqrt{\langle p_1, .., p_n \rangle}$), contradicts $\bigwedge_{i=1}^{n} p_i = 0 \wedge q \neq 0$. The $k, f_1, \ldots, f_n$ form a "Nullstellensatz refutation", after Hilbert's theorem.

Theorem 2 (Hilbert's Nullstellensatz [Hil93])**.** *Let K be a field and L an algebraically closed extension of K, and $q, p_1, \ldots, p_n \in K[x_1, \ldots, x_m]$, then the following are equivalent:*

i) $\forall z \in L^m. \bigwedge_{i=1}^{n} p_i(z) = 0 \rightarrow q(z) = 0$

ii) $q \in \sqrt{\langle p_1, \ldots, p_n \rangle}$

Note that Theorem 2 guarantees the equivalence for algebraically closed fields. In our application, we only need the trivial direction, which holds for arbitrary integral domains. We obtain a Nullstellensatz-certificate using extended Gröbner bases [BWK93, §6.2,§5.6] or [Mis93, §3.4]. We do not explain the construction of such certificates, since this is out of the scope of this thesis, and refer the interested reader to [Mis93, BWK93]. We obtain the certificates from an implementation in ML, or from a computer algebra system (e.g. Singular or M2).

A reflected certificate-checker

We reflect Nullstellensatz certificates by $\langle \boldsymbol{\rho} \rangle$ and $(\!|.|\!)_{\langle \boldsymbol{\rho} \rangle}$ below:

$$\begin{array}{lll} \mathsf{datatype}\ \langle \boldsymbol{\rho} \rangle & = \boldsymbol{p}_{\mathbb{N}} | \langle \boldsymbol{\rho} \rangle + \langle \boldsymbol{\rho} \rangle | \boldsymbol{\rho} * \langle \boldsymbol{\rho} \rangle & (\!|\boldsymbol{p}_n|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} = ps!n \\ & & (\!|p * c|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} = p \circledast (\!|c|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} \\ & & (\!|c + d|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} = (\!|c|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} \oplus (\!|d|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle} \end{array}$$

The idea is that given normalised polynomials $ps = [p_1, \ldots, p_n]$ and a certificate c, $(\!|c|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle}$ evaluates to a polynomial in the ideal $\langle p_1, \ldots, p_n \rangle$. Clearly if $p_1, \ldots, p_n$ are as in Theorem 2 and $(\!|c|\!)^{ps}_{\langle \boldsymbol{\rho} \rangle}$ evaluates to a power of q then $q \in \sqrt{\langle p_1, \ldots, p_n \rangle}$ yielding a contradiction, cf. (3.31). We use $\boldsymbol{p}_n$ to refer to the n^{th} polynomial in ps. Of course a well-formed certificate must not contain references outside ps, cf. $\mathsf{wf}_{\langle \boldsymbol{\rho} \rangle}$ below. Well-formed certificates satisfy the main property (3.31).

[1] This terminology is also used in e.g. [BW01]

$$\begin{array}{l}\mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ \boldsymbol{p}_n\ k \quad = n < k\\ \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ (p * c)\ k = \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ k\\ \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ (c + d)\ k \quad = \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ k \wedge \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ d\ k\\ \qquad\qquad \textbf{(in } idom\textbf{)}\ (\!|c|\!)^{ps}_{\langle\boldsymbol{\rho}\rangle} = (q_1 \circledast \ldots \circledast q_n)^{\downarrow\boldsymbol{m}} \wedge \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ |ps|\\ \rightarrow (\forall p \in \{\!\{ps\}\!\}.(\!|p|\!)^e_{\boldsymbol{\rho}} = 0 \wedge \forall q \in \{\!\{[q_1,\ldots,q_n]\}\!\}.(\!|q|\!)^e_{\boldsymbol{\rho}} \neq 0) \leftrightarrow \mathit{False}\end{array} \tag{3.31}$$

Note that the conditions of (3.31) are executable and correspond to the function $\mathsf{check}_{\langle\boldsymbol{\rho}\rangle}$, which satisfies (3.32). Note that $\mathsf{check}_{\langle\boldsymbol{\rho}\rangle}$ reflects a checker for radical ideal membership.

$$\begin{array}{l}\mathsf{check}_{\langle\boldsymbol{\rho}\rangle}\ ps\ [q_1,\ldots,q_n]\ c\ m = \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ |ps| \wedge (\!|c|\!)^{ps}_{\langle\boldsymbol{\rho}\rangle} = (q_1 \circledast \ldots \circledast q_n)^{\downarrow\boldsymbol{m}}\\ \textbf{(in } idom\textbf{)}\ \mathsf{check}_{\langle\boldsymbol{\rho}\rangle}\ \overline{p}\ \overline{q}\ c\ m \rightarrow (\forall p \in \{\!\{\overline{p}\}\!\}.(\!|p|\!)^e_{\boldsymbol{\rho}} = 0 \wedge \forall q \in \{\!\{\overline{q}\}\!\}.(\!|q|\!)^e_{\boldsymbol{\rho}} \neq 0) \leftrightarrow \mathit{False}\end{array} \tag{3.32}$$

3.2.2 An interesting subset of the ∀∃-fragment

We present a proof-procedure for an interesting subset of ∀∃ ring problems, which generalises solving systems of linear equations with polynomial coefficients. This presentation is based on [Har07a], but we discuss some connections to [BWK93]. The goal is to prove statements of the form (3.b) or, equivalently, using matrix and vectors notation (3.c), where $e_i(\vec{x})$, $p_{ij}(\vec{x})$ and $a_i(\vec{x})$ are elements of $R[x_1,\ldots,x_n]$ and R is a commutative ring.

$$\forall x_1..x_n.\bigwedge_{i=1}^{l} e_i(x_1,..,x_n) = 0 \rightarrow \exists y_1..y_m.\bigwedge_{i=1}^{k}\sum_{j=1}^{m} p_{ij}(x_1,..,x_n)\cdot y_j = a_i(x_1,..,x_n) \tag{3.b}$$

$$\forall \vec{x}.\vec{e}(\vec{x}) = \vec{0} \rightarrow \exists \vec{y}.P(\vec{x})\vec{y} = \vec{a}(\vec{x}) \tag{3.c}$$

Internally we use again the extended Gröbner bases algorithm to find an ideal membership certificate $r_1(\vec{x},\vec{u}),..,r_n(\vec{x},\vec{u}),q_1(\vec{x},\vec{u}),..,q_m(\vec{x},\vec{u})$ for the problem in (3.d).

$$\vec{a}(\vec{x}) \bullet \vec{u} \in \langle e_1,..,e_n,P^1(\vec{x}) \bullet \vec{u},..,P^m(\vec{x}) \bullet \vec{u}\rangle, \tag{3.d}$$

In (3.d), $\vec{u} = (u_1,\ldots,u_k)$ is a fresh vector of variables, $P^j(\vec{x})$ is the j^{th} column of $P(\vec{x})$ and $\bullet$ denotes the inner product of vectors, i.e. $\vec{a} \bullet \vec{b} = \sum_{i=1}^{t} a_i \cdot b_i$. Concretely, we have

$$\vec{a}(\vec{x}) \bullet \vec{u} = \vec{e}(\vec{x}) \bullet \vec{r}(\vec{x},\vec{u}) + \sum_{i=1}^{m}(P^i(\vec{x}) \bullet \vec{u}) \cdot q_i(\vec{x}).$$

First note that on the left hand side each monomial contains exactly one u_i. Next we decompose the $q_i(\vec{x},\vec{u})$ and $r_i(\vec{x},\vec{u})$ into $q_i(\vec{x},\vec{u}) = c_i(\vec{x}) + d_i(\vec{x},\vec{u})$ and $r_i(\vec{x},\vec{u}) = s_i(\vec{x},\vec{u}) + t_i(\vec{x},\vec{u})$, such that:

1. $c_i(\vec{x})$ does not involve any u_j,
2. all monomials in $d_i(\vec{x},\vec{u})$ contain at least one u_j,
3. each monomial in $s_i(\vec{x},\vec{u})$ has degree 1 in exactly one u_j, and
4. each monomial in $t_i(\vec{x},\vec{u})$ either does not involve any u_i, involves more than one or has degree > 1 in one of them.

Now according to the observation above about the degree of u_i in the LHS, all terms on the RHS not of that form have to cancel, leaving:

$$\vec{a}(\vec{x}) \bullet \vec{u} = \vec{e}(\vec{x}) \bullet \vec{s}(\vec{x},\vec{u}) + \sum_{i=1}^{m} P^i(\vec{x}) \bullet \vec{u} \cdot c_i(\vec{x})$$

Now using the fact that $\vec{e} = \vec{0}$, and successively setting $u_i = 1$ and $u_j = 0$, for $j \neq i, 1 \leq i,j \leq k$, yields k equations for $1 \leq i \leq k$:

$$a_i(\vec{x}) = c_1(\vec{x}) \cdot p_{i1}(\vec{x}) + \cdots + c_m(\vec{x}) \cdot p_{im}(\vec{x}).$$

The vector $\vec{c}(\vec{x})$ is therefore a witness for $\vec{y}$.

An interpretation into modules

The problem (3.b), solved in [Har07a], is a slight generalisation of solving inhomogeneous linear equations with polynomial coefficients, i.e. (3.b) without the assumption part. This problem is often presented in the Gröbner bases literature as an application to modules and Syzygies computation. We slightly generalise the presentations in [BWK93, Mis93] to deal with (3.b) and establish hence a connection to [Har07a]. In particular we emphasise the contributions of [Har07a]. I am thankful to John Harrison for encouraging me to include this discussion here.

Given a ring R, an abelian group M and a mapping $(r \in R, x \in M) \mapsto r \circ x$, then M is an R module if for all $x, y \in M$ and $s, t \in R$ the following holds:

$$s \circ (x + y) = s \circ x + s \circ y \wedge (s + t) \circ x = s \circ x + t \circ x$$
$$(s \cdot t) \circ x = s \circ (t \circ x) \wedge 1 \circ x = x.$$

This is a generalisation of vector spaces from fields to rings. It is also a generalisation of ideals: any ideal I of R is an R-module, where $\circ = \cdot$. If for $N = \{x_i\}_{i \in J} \subset M$ and any $x \in M$ there are $s_i \in R, i \in J$ such that $x = \sum_{i \in J} s_i \circ x_i$, then M is generated by N. Let $\{x_1, \ldots, x_n\} \subseteq M$, then any n-tuple $(r_1, \ldots, r_n)$ satisfying $\sum_{i=1}^{n} r_i \circ x_i = 0$ is a Syzygy of $(x_1, \ldots, x_n)$. The set of all Syzygies of $(x_1, \ldots, x_n)$ forms an R-module. It is a sub-module of R^n, the set of all n-tuples over R with the usual operations.

Consider an instance of (3.b) and assume $\bigwedge_{i=1}^{l} e_i(\vec{x}) = 0$ and let $E = \langle e_1(\vec{x}), \ldots, e_l(\vec{x}) \rangle$. First note that we can safely assume $l = k$, for if $k < l$ then we can just duplicate some equations under $\exists$ and conversely if $l < k$. Our original problem amounts to decide if the inhomogeneous system of linear equations

$$\begin{aligned} p_{11} \cdot y_1 + \cdots + p_{1m} \cdot y_m &= a_1 \\ &\vdots \\ p_{k1} \cdot y_1 + \cdots + p_{km} \cdot y_m &= a_k \end{aligned} \tag{3.e}$$

has a solution in the unknowns $y_1, \ldots, y_m$. We can view any solution as an element of $R[\vec{x}]^m$, which is an $R[\vec{x}]$ module and hence (3.e) is equivalent to (3.f).

$$\vec{p_1} \cdot y_1 + \cdots + \vec{p_m} \cdot y_m = \vec{a} \tag{3.f}$$

Such a solution exists if and only if

$$\vec{a} \in lin(\vec{p_1}, \ldots, \vec{p_m}) + E^k, \tag{3.g}$$

where $lin(\vec{p_1}, \ldots, \vec{p_m})$ is the linear span of $\vec{p_1}, \ldots, \vec{p_m}$ in the module $R[\vec{x}]^k$.

Now let $u_1, \ldots, u_k$ be k new variables and consider

$$H_1(R[\vec{x}, \vec{u}]) = \{h_1 \cdot u_1 + \cdots + h_k \cdot u_k | h_1, \ldots, h_k \in R[\vec{x}]\}.$$

$H_1(R[\vec{x}, \vec{u}])$ is a $R[\vec{x}]$ module and is isomorphic to $R[\vec{x}]^k$ under

$$\begin{aligned} \varphi : \quad R[\vec{x}]^k &\to H_1(R[\vec{x}, \vec{u}]) \\ (h_1, \ldots, h_k) &\mapsto h_1 \cdot u_1 + \cdots + h_k \cdot u_k \end{aligned}$$

Hence (3.g) holds if and only if $\varphi(\vec{a}) \in lin(\varphi(\vec{p_1}), \ldots, \varphi(\vec{p_m})) + \varphi(E^k)$, which can be decided using Gröbner bases as an ideal membership problem:

$$\varphi(\vec{a}) \in \langle \varphi(\pi_1(\vec{e})), \ldots, \varphi(\pi_l(\vec{e})), \varphi(\vec{p_1}), \ldots, \varphi(\vec{p_m}) \rangle,$$

where $\pi_j(\vec{e})$ is the j^{th} rotation of $\vec{e}$'s elements, i.e. $\pi_j(\vec{e}) = (e_j, \ldots, e_l, e_1, \ldots, e_{j-1})$, for every $1 \leq j \leq l$. Note that $\pi_1(\vec{e}), \ldots, \pi_l(\vec{e})$ generate $E^l = E^k$. Note that this is just a paraphrase of (3.d). In [BWK93] the authors note that it is not necessary to build a *complete* Gröbner

basis of the ideal but just a so called 1-Gröbner basis, where during construction only S-polynomials with degree *at most* 1 in any of the u_i's are included. Of course computing 1-Gröbner bases is much more efficient. In [Har07a] the author computes a complete Gröbner basis but then argues that the terms leading to polynomials with degree greater than 1 in any of the u_i's must vanish.

The presentation in [BWK93] does not deal with assumptions, the simple adaptation above is ours. Most important is that in [BWK93] the authors look for a solution over $R[\vec{x}]$, whereas [Har07a] is interested in a solution over R. The main result in [Har07a] is that existence of solutions in R is equivalent to existence of solutions in $R[\vec{x}]$, as long as we regard the statement as true over *all* rings. See [Har07a] for a very beautiful proof-theoretical proof using the Horn-Herbrand theorem.

The approach in [BWK93, Mis93] to compute a Syzygy basis for the inhomogeneous equations yields a finite representation of *all* solutions to the linear system. This is irrelevant in a theorem proving context since we just want to prove the *existence* of solutions. One witness is therefore enough.

3.2.3 Integration

An integration of the previous procedures using $\mathsf{check}_{\langle\rho\rangle}$ in §3.2.1 is simple. We present here a context-sensitive integration, which allows to use the previous procedures in semi-rings ($\mathbb{N}$ is the main application). The core of the development is a normaliser for polynomial expressions.

The normaliser is a conversion, parametrised with all the underlying operations and corresponding theorems, e.g. associativity, commutativity and distributivity. The normaliser deals with polynomials over semi-rings, rings and fields, depending on how many operations and axioms we fill in. The normaliser is also parametrised by the following four functions, needed to recognise and compute on the coefficients:
is_const: $term \to bool$ recognises terms as numeral constants,
dest_const: $term \to \mathbb{Q}$ obtains a rational number out of a numeral constant term,
mk_const: $\mathbb{Q} \to term$ obtains a term representing a rational number, and
num_cv: $term \to thm$ a normalisation *conversion* of numeral expressions

The parameters to an *instance* of the procedure are mainly determined by those above of the normaliser. The functions above, *parametrised by an extra morphism*, form the non-logical part. The logical part consists of a large number of theorems (the input to the normaliser) and the axioms of *gb* below. We add these to the context data by fact declaration in *gb*. In particular, when morphed, we pass all parameters to the normaliser. In order to manage the several instances of *gb*, the context data consists of a mapping of keys (morphed locale predicates of the different instances) to the *instance* data above.

$$\begin{aligned}\textbf{locale } gb = semiring + \quad & \textbf{assumes } x + y = x + z \leftrightarrow y = y \\ & \textbf{and } w \cdot y + x \cdot z = w \cdot z + x \cdot y \leftrightarrow w = x \vee y = z\end{aligned}$$

Overview: when invoked on a problem P, the method extracts a polynomial q occurring in P. Using q we search for an instance installed in the context data and then proceed as described in §3.2.1. We use an extended Gröbner bases algorithm to find a refutation for each disjunct of P's DNF. Thereby, we treat an equation $p = q$ in P, as if it were $p - q = 0$. When interpreting certificates, we split them into a "positive" and a "negative" part and obtain a contradiction in similar manner to §3.2.1.

To develop the proof-method for both universal and $\forall\exists$ problems, we needed round 400 lines of Isabelle proofs and 1800 lines of SML code.

Applications of §3.2.1 include simple algebraic and geometric properties such as solving the quartic equation, Simson's theorem etc. The procedure in §3.2.2 is useful to prove interesting number theoretic statements. In particular it deals with divisibility, coprime and equality modulo. A prominent example is the Chinese remainder (for a specific number of moduli) below. See [Har07a] for other interesting examples.

$$\forall a, b, u, v.\mathsf{coprime}\ a\ b \rightarrow \exists x.x \equiv u\ (mod\ a) \wedge x \equiv v\ (mod\ b).$$

3.3 Inequalities via sums of squares

After considering equations and disequations, we present here a method to prove universal formulae with polynomial inequalities (i.e. $p \geq 0$ for a polynomial p) as atoms. We consider polynomials over $\mathbb{R}$ with coefficients in $\mathbb{Q}$. For real polynomials, using the Nullstellensatz turns quite unsatisfactory, e.g. $x^2 + 1$ has no solution over $\mathbb{R}$ but $1 \notin \sqrt{\langle x^2 + 1\rangle}$. Note that the Nullstellensatz only guarantees equivalence if the underlying field is algebraically closed, which is not the case for $\mathbb{R}$. There is $\mathbb{R}$-analogue of Theorem 2:

Theorem 3 (Real Nullstellensatz)**.** *Let $p_1(\vec{x}) = 0, \ldots, p_n(\vec{x}) = 0$ for $\vec{x} \in \mathbb{R}^m$ be polynomial equations, then the following are equivalent:*

i) $\neg\exists \vec{z} \in \mathbb{R}^m. \bigwedge_{i=1}^{n} p_i(\vec{z}) = 0$

ii) There exist $s_1(\vec{x}), \ldots, s_k(\vec{x}) \in \mathbb{R}[x_1, \ldots, x_n]$ *s.t.* $1 + \sum_{i=1}^{k} s_i(\vec{x})^2 \in \langle p_1, \ldots, p_n\rangle$

This is indeed a special instance of the more general Positivstellensatz by Stengle. Recall that given a ring R and $S \subseteq R$, $Cone(S)$ is the sub-semiring generated by S and all squares of R, i.e. $\{a^2 | a \in R\}$.

Theorem 4 (Stengle's Postivstellensatz [Ste74])**.** *Consider the $n + m + t$ real polynomials $p_1, \ldots, p_n$, $q_1, \ldots, q_m$ and $r_1, \ldots, r_t$, then the following are equivalent:*

i) $\neg\exists \vec{z}. \bigwedge_{i=1}^{n} p_i(\vec{z}) = 0 \wedge \bigwedge_{i=1}^{m} q_i(\vec{z}) \geq 0 \wedge \bigwedge_{i=1}^{t} r_i(\vec{z}) \neq 0$

ii) There exists $p \in \langle p_1, \ldots, p_n\rangle, q \in Cone(q_1, \ldots, q_m)$ *and* $r = \prod_{i=1}^{t} r_i^{k_i}$, *s.t.* $p + q + r^2 = 0$

The proofs of Theorem 3 and Theorem 4 are far out of the scope of this thesis, but we refer the interested reader to [Ste74, BPR03]. We present in the following how to adapt a method pioneered by Parillo [Par03] to obtain Positivstellensatz certificates (p, q and r in Theorem 4) using semi-definite programming (SDP).

In the following we first consider just one real polynomial $p \in \mathbb{R}[x_1, \ldots, x_n]$ and present known results on the relation between p being positive semi-definite (PSD), i.e. $\forall \vec{x} \in \mathbb{R}^n.p(\vec{x}) \geq 0$, and p being a sum of squares (SOS), i.e. $\forall \vec{x} \in \mathbb{R}^n.p(\vec{x}) = \sum_{i=1}^{k} s_i(\vec{x})^2$ for some polynomials $s_1, \ldots, s_k \in \mathbb{R}[x_1, \ldots, x_n]$. Trivially SOS implies PSD.

3.3.1 SOS, PSD and Hilbert's theorem

Hilbert [Hil88] studied the relation between PSD and SOS and raised the problem if a PSD polynomial p always has a SOS decomposition. He proved that this is not true if we restrict the s_i's above to be polynomials by giving a rather complicated counterexample. Motzkin found a much simpler one: $M(x, y, z) = x^4 \cdot y^2 + x^2 \cdot y^4 + z^6 - 3 \cdot x^2 \cdot y^2 \cdot z^2$. Hilbert included the PSD problem in his famous list (the 17^{th} problem) asking for an SOS decomposition of rational functions. Artin [Art27] solved this problem. Unfortunately, his proof does not yield an algorithm to construct the rational functions. Hilbert also studied the relation between PSD and SOS in special cases, and proved interesting results. A polynomial p is an m-form (or homogeneous) if all its monomials have the same total degree m. Let $P_{n,m}$ be the set of PSD m-forms in n variables and $\Sigma_{n,m}$ the set of m-forms $p = \sum_i h_i^2$, where h_i are $\frac{m}{2}$-forms in n variables, then

Theorem 5 (Hilbert [Hil88])**.** $\Sigma_{n,m} \subseteq P_{n,m}$ *and the equality holds* only *for*

a) Bivariate forms: $n = 2$

b) Quadratic forms: $m = 2$
c) Ternary quartics: $n = 3$ *and* $m = 4$.

In terms of polynomials, instead of forms (using de-homogenisation), the cases in Theorem 5 correspond to univariate, quadratic and quartic (in three variables) polynomials respectively.

3.3.2 Quadratic forms and SOS via SDP

Quadratic forms

Quadratic forms play a particularly important role in our application. If p is a quadratic form, then we can represent it in a standard way

$$p(x_1, \ldots, x_n) = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \cdot x_i \cdot x_j = \vec{x}^T A_p \vec{x}, \text{ where } A_p = (a)_{ij} \text{ and } a_{ij} = a_{ji}.$$

Note that if in some representation $a_{ij} \neq a_{ji}$ then we can set $a'_{ij} = a'_{ji} = \frac{a_{ij}+a_{ji}}{2}$. Hence a PSD polynomial p is represented by a symmetric matrix $A_p = (a)_{ij}$. Generally we say that a symmetric matrix A is PSD if $\forall \vec{x}.\vec{x}^T A\vec{x} \geq 0$, which is equivalent to saying that the associated polynomial is PSD. Other equivalent formulations of A being PSD is that all its eigenvalues are positive (they are all real since A is symmetric) or that $A = L^T L$ for some triangular matrix L or also equivalently that $A = LDL^T$ for some lower-triangular matrix L and a diagonal matrix D. The LDL^T formulation is more appealing for symbolic applications since computing it only needs rational operations, whereas square roots are needed to obtain $L^T L$ using Choleski decomposition.

Finding SOS decompositions via SDP

The key idea is to view a given polynomial $p(x_1, \ldots, x_n)$ as a polynomial in new variables $z_1, \ldots, z_m$ suitably chosen such that $p(z_1, \ldots, z_m)$ is a quadratic form.

First note that if p is PSD then it must have an even degree $2d$. The only monomials in $\mathbb{R}[x_1, \ldots, x_n]$ that could appear in an SOS decomposition of $p(x_1, \ldots, x_n)$ are those with degree at most d, since the others, when squared, would not cancel. There are $m = \binom{n+d}{d}$ many of these and let $z_1, \ldots, z_m$ denote them. We can see $p(x_1, \ldots, x_n)$ as a polynomial in $z_1, \ldots, z_m$ and let us express p as quadratic form in $z_1, \ldots, z_m$, i.e. we ask for a matrix A such that:

$$p(\vec{x}) = \vec{z}^T A\vec{z} = \sum_{i=1}^{m} \sum_{i=1}^{m} a_{ij} z_i z_j. \tag{3.h}$$

Clearly if A is PSD then so is p. Note that since $z_1, \ldots, z_m$ are not algebraically independent, A is not unique and hence might not be PSD for some representations. By expanding both sides of (3.h) and comparing coefficients we obtain a set of linear constraints on the coefficients a_{ij} of A. Clearly we are only looking for matrices satisfying these constraints. We can further show that the set of all such matrices forms an affine subspace. Semidefinite programming solves exactly this problem: it finds a PSD matrix subject to some linear constraints (it actually maximises a linear function).

Now assume A is PSD. Then we have $A = L^T L$ for a lower-triangular matrix L by Cholesky decomposition. Hence we can "read off" a sums of squares decomposition since $p(\vec{x}) = \sum_{i=1}^{m}((L\vec{z})_i)^2$.

A simple Example [Par03]

Let $p(x, y) = 2x^4 + 2x^3y - x^2y^2 + 5y^4$. Then no monomial of degree larger than 2 can appear in the SOS decomposition and in fact the only candidates are $z_1 = x^2, z_2 = y^2$ and $z_2 = xy$. Now writing $p(x, y) = \vec{z}^T A\vec{z}$ and expanding we obtain $p(x, y) = a_{11}x^4 + a_{22}y^4 +$

$(a_{33} + 2a_{12})x^2y^2 + 2a_{13}x^3y + 2a_{23}xy^3$. Remember that A is symmetric. Consequently the following must hold:

$$a_{11} = 2, \quad a_{22} = 5, \quad a_{33} + 2a_{12} = -1, \quad 2a_{13} = 2, \quad 2a_{23} = 0$$

Using SDP we find a particular PSD A

$$A = \begin{pmatrix} 2 & -3 & 1 \\ -3 & 5 & 0 \\ 1 & 0 & 5 \end{pmatrix} = L^T L, \text{ for } L = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & -3 & 1 \\ 0 & 1 & 3 \end{pmatrix}.$$

Now we can "read off" the SOS decomposition: $p(x, y) = \frac{1}{2}(2x^2 - 3y^2 + xy)^2 + \frac{1}{2}(y^2 + 3xy)^2$. Strictly speaking this is not an SOS decomposition, but every positive rational number is the sum of four rational squares.

3.3.3 Finding Positivstellensatz certificates

Parrilo proved that we can use SDP to find the refutations in Theorem 4.

Theorem 6 (Parrilo [Par00])**.** *Consider a system of equations and inequalities as in Theorem 4 i). Then the search for bounded degree Positivstellensatz refutation can be done using semidefinite programming. If the degree bound is chosen to be large enough, then the SDPs will be feasible, and the certificates obtained from its solution.*

Proof. Given a fixed degree d and let $r = \prod_{i=1}^{t} r_i^{2m}$, where m is such that $\partial r \geq d$. For the inequalities choose $d_2 \geq \max\ d\ \partial r$ and write

$$q = f_0 + f_1 q_1 + \cdots + f_m q_m + f_{12} q_1 q_2 + \cdots + f_{12\ldots m} q_1 \ldots q_m,$$

and give a parametrisation of the f_i's with degree at most d_2. Do similarly for $p \in \langle p_1, \ldots, p_n \rangle$:

$$p = \sum_{i=1}^{n} g_i p_i,$$

and parametrise the g_i's with degree at most d_2. Now solve the SDP for f_i are sums of squares, subject to the equations implied by $p + q + r^2 = 0$. The decision variables are the coefficients of the f_i's and g_i's.

The last part of the theorem follows from the existence of such certificates by Theorem 4 and the fact that the SDP is feasible for all $d \geq d_0$ for some d_0. □

3.3.4 Integration

Unfortunately we have not been able to finish an integration of the procedure on time. We formalise a certificate checker and prove it correct below, but it remains to implement the interaction with SDP solvers. The formalisation of the checker needed 180 lines of Isabelle proofs, of course not counting the formalisation of multivariate polynomials in§3.1.2.

A reflected checker

We reflect cone membership certificates by $\boldsymbol{\rho}_{\trianglerighteq}$ and $(\!|.|\!)_{\boldsymbol{\rho}_{\trianglerighteq}}$ below:

$$\begin{array}{lll} & \mathsf{datatype}\ \boldsymbol{\rho}_{\trianglerighteq} & = \boldsymbol{p}_{\mathbb{N}} | \boldsymbol{\rho^2} | \boldsymbol{\rho}_{\trianglerighteq} + \boldsymbol{\rho}_{\trianglerighteq} | \boldsymbol{\rho}_{\trianglerighteq} * \boldsymbol{\rho}_{\trianglerighteq} | \boldsymbol{\rho}_{\trianglerighteq}{}^{\mathbb{N}} \\ (\!|\boldsymbol{p}_n|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} & = qs!n & \\ (\!|c * d|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} & = (\!|c|\!)^{ps}_{\boldsymbol{\rho}_{\trianglerighteq}} \circledast (\!|d|\!)^{ps}_{\boldsymbol{\rho}_{\trianglerighteq}} & (\!|c + d|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} = (\!|c|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} \oplus (\!|d|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} \\ (\!|p^{\boldsymbol{n}}|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} & = (\!|p|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} \downarrow^{\boldsymbol{n}} & (\!|p^{\boldsymbol{2}}|\!)^{qs}_{\boldsymbol{\rho}_{\trianglerighteq}} = (\!|p|\!)^{qs}_{\boldsymbol{\rho}} \circledast (\!|p|\!)^{qs}_{\boldsymbol{\rho}} \end{array}$$

A Positivstellensatz certificate consists of ideal membership and a cone membership certificate, and of the powers $k_1, \ldots, k_t$ for the disequations (cf. Theorem 4). The cone membership

certificate and the ideal membership certificate must be well-formed ($\mathsf{wf}_{\boldsymbol{\rho}_{\mathbb{>}}}$ is analogous to $\mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}$ in §3.2.1). Well-formed certificates satisfy the following "checker version" of Theorem 4:

$$\begin{array}{r}(\!|c|\!)^{ps}_{\langle\boldsymbol{\rho}\rangle} \oplus (\!|d|\!)^{qs}_{\boldsymbol{\rho}_{\mathbb{>}}} \oplus (r_1{}^{\downarrow \boldsymbol{k_1}} \circledast \ldots \circledast r_p{}^{\downarrow \boldsymbol{k_p}}) = \widehat{\mathbf{0}_{\boldsymbol{r}}} \wedge \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ |ps| \wedge \mathsf{wf}_{\boldsymbol{\rho}_{\mathbb{>}}}\ d\ |qs| \\ \rightarrow (\forall p \in \{\!\{ps\}\!\}.(\!|p|\!)^e_{\boldsymbol{\rho}} = 0 \wedge \forall q \in \{\!\{qs\}\!\}.(\!|q|\!)^e_{\boldsymbol{\rho}} \geq 0 \wedge \forall r \in \{\!\{rs\}\!\}.(\!|r|\!)^e_{\boldsymbol{\rho}} \neq 0) \leftrightarrow \mathit{False}\end{array} \tag{3.33}$$

The proof is similar to (3.31), but we have done it only for axiomatic type classes [Wen97], and not inside locales. There are some problems when using locales for ordered structures in Isabelle, e.g. monotony theorems need locales with more than one ordering. Analogously to §3.2.1, we define an executable checker $\mathsf{check}_{\boldsymbol{\rho}_{\mathbb{>}}}$ and prove (3.34).

$$\begin{array}{r}\mathsf{check}_{\boldsymbol{\rho}_{\mathbb{>}}}\ ps\ qs\ [r_1, \ldots, r_p]\ c\ d\ [k_1, \ldots, k_p] = \\ \mathsf{wf}_{\langle\boldsymbol{\rho}\rangle}\ c\ |ps| \wedge \mathsf{wf}_{\boldsymbol{\rho}_{\mathbb{>}}}\ d\ |qs| \wedge (\!|c|\!)^{ps}_{\langle\boldsymbol{\rho}\rangle} \oplus (\!|d|\!)^{qs}_{\boldsymbol{\rho}_{\mathbb{>}}} \oplus (r_1{}^{\downarrow \boldsymbol{k_1}} \circledast \ldots \circledast r_p{}^{\downarrow \boldsymbol{k_p}})\end{array}$$

$$\begin{array}{r}\mathsf{check}_{\boldsymbol{\rho}_{\mathbb{>}}}\ ps\ qs\ rs\ c\ d\ ks \rightarrow \\ (\forall p \in \{\!\{ps\}\!\}.(\!|p|\!)^e_{\boldsymbol{\rho}} = 0 \wedge \forall q \in \{\!\{qs\}\!\}.(\!|q|\!)^e_{\boldsymbol{\rho}} \geq 0 \wedge \forall r \in \{\!\{rs\}\!\}.(\!|r|\!)^e_{\boldsymbol{\rho}} \neq 0) \leftrightarrow \mathit{False}\end{array} \tag{3.34}$$

3.4 Related work

Formalising polynomials as functions (as in §3.1) has been done quite early in HOL for $\mathbb{R}$, but there, due to the lack of classes or equivalent abstract specification mechanisms, had to be duplicated for $\mathbb{C}$ or other structures. Our formalisation in §3.1 is mainly a generalisation to classes. In [Mah06b] there is an alternative formalisation of univariate polynomials in Coq. Another alternative formalisation of multivariate polynomials using dependent types in Coq is presented in [GM05]. The formalisation in [GM05] does not include e.g. pseudo-division or a proof of the uniqueness property. See also [Bal99] for a nice mathematical formalisation of polynomials and several interesting properties. The formalisation in [Bal99] is not executable.

The membership problem for polynomials ideals (PIMP) is central for §3.2. Given polynomials $p_1, \ldots, p_n$ and f, find polynomials $q_1, \ldots, q_n$ such that $f = \sum_{i=1}^{n} p_i \cdot q_i$, if $f \in \langle p_1, \ldots, p_n \rangle$. In her pioneer 1926 paper [Her26], Herman proved this problem decidable and gave a doubly exponential upper bound on the degree of $q_1, \ldots, q_n$. See also [Sei74, Ric74] for decidability of PIMP for more general ideals. In [MM82], Mayr and Meyer show that the doubly exponential growth above is unavoidable in general. This paper plays a crucial role in lower bound complexity results for decision problems in algebraically and real closed fields in Chapter 4. See also [Bro87, Kol88] for single exponential upper bounds on the degrees of $q_1, \ldots, q_n$ in the special case of $f = 1$. Note that radical ideal membership is not more complex than PIMP using the trick of Rabinovitsch. Using Gröbner bases for PIMP yields doubly exponential degree in worst case, cf. [Huy86]. See [May97] for a nice survey of more complexity results. For more details on Gröbner bases see [Buc65] or the seminal books [BWK93, Mis93]. See also [Bün91, BN98] for a view from term rewriting and the connection to the Knuth-Bendix completion procedure. Our application in §3.2.1 to Hilbert's Nullstellensatz is quite standard in the literature, but see [BWK93, §6.4] for an interesting section on uniform word problems (cf. also [Har08, §5.10]). In geometry, Gröbner bases also serve as a decision method for several geometrical problems, see [Kap96, CG01] for an interesting comparative survey. The methods presented in §3.2 are implemented in HOL Light, but, to our knowledge, in no other theorem prover, except Isabelle.

The SDP vs. SOS problem has been raised and studied by Hilbert [Hil88]. Artin's proof of Hilbert's 17^{th} problem does not yield a method to construct the rational functions in the SOS decomposition. There is a large series of Positivstellensätze by Pòlya [Pól28], Stengle [Ste74], Schmüdgen [Sch91] etc, but they all do not yield an algorithmic construction of the SOS decomposition, except [Pól28], but see [Sch99] for algorithms obtained from different proofs of these theorems. Algorithms to compute an SOS decomposition using the Gram matrix

method appear in [CLR95, PW98]. Using SDP for Positivstellensatz certificates seems to be first introduced in [Par00, Par03]. There is a software to solve SOS problems [PPSP04] based on MATLAB, but we recently discovered an open source and free implementation [PP07] on top of Macaulay. The first integration into an LCF-like theorem prover appears in [Har07b]. This implementations has also been ported to Coq.

Chapter 4

Elimination of quantifiers

> The method is extremely valuable when we want to beat a particular theory into the ground [...] The method may be thought of as a direct attack on a theory.
>
> *(Chang and Keisler)*

Contents

The method of quantifier elimination (qe. for short) is a powerful tool of model theory and the decision problem and is the focus of this chapter. We start with some preliminary material in §4.1 and then present quantifier elimination procedures for several theories. In §4.2 we reconsider dense linear order without endpoints and give an alternative approach to Langford's procedure in §2.1.3. In §4.3 we consider linear arithmetic over three theories: parametric linear problems over ordered fields in §4.3.1, Presburger arithmetic in §4.3.2 and mixed real-integer linear arithmetic in §4.3.3. In §4.4 and §4.5 we present qe. for algebraically and real closed fields.

4.1 Preliminaries

The qe. problem for an L-theory T asks if any L-formula is T-equivalent to a quantifier free (qf. for short) L-formula. If T admits qe. then T is model complete and also substructure complete. Moreover if ground L-atoms are decidable then T is decidable. Theories admitting qe. are very rare and of considerable interest. The following theorem gives an impression on the strength of the qe. property for algebraic theories, see e.g. [MMvdD83] for similar results.

Theorem 7 (Van den Dries 1980 [vdD80])**.** *A linearly ordered ring whose theory admits elimination of quantifiers is a real closed field*

In principle every theory T has a conservative extension T' which admits qe., by adding new relation symbols R_ϕ, also called a Skolem-relation for ϕ, and new axioms $\forall x.\phi(x) \leftrightarrow R_\phi(x)$ for every formula ϕ of L. This extension was already used by Skolem in [Sko70a], but is now known as Morley-extension. In this thesis we are interested in theories admitting qe., for which we have an *algorithm*, i.e. a quantifier elimination procedure (qep. for short), which given any L-formula ϕ returns a T-equivalent qf. L-formula. Note that it is sufficient to have a procedure to eliminate only one existential quantifier from $\exists x.P(x)$, where P is quantifier free. By applying this procedure recursively to the innermost quantifier we obtain a qep. for the whole theory. This can be further relaxed to P being a conjunction of atoms all of them containing the bound variable x. Consider a qf. P and transform it into DNF,

i.e. $P(x) \leftrightarrow \bigvee_i \bigwedge_j a_{ij}(x) \wedge \bigwedge_j c_{ij}$, then $\exists x.P(x) \leftrightarrow \bigvee_i \bigwedge_j c_{ij} \wedge \exists x. \bigwedge_j a_{ij}(x)$, where x occurs in the atoms $a_{ij}(x)$ but not in c_{ij}. Remember that $\exists$ satisfies:

$$(\exists x.P(x) \vee Q(x)) \leftrightarrow ((\exists x.P(x)) \vee (\exists x.Q(x))) \tag{4.i}$$

$$(\exists x.P \wedge Q(x)) \leftrightarrow (P \wedge \exists x.Q(x)) \tag{4.j}$$

This "lifting" property can be generalised to any *representative* subset of qf. formulae, where a subset of qf. formulae Φ is *representative* if any qf. formula ϕ has a qf. T-equivalent formula $\psi \in \Phi$, which can be constructed by an *algorithm*.

Elimination sets and virtual substitution

Weispfenning introduced an interesting generic method for qe. and applied it successfully to several non-trivial examples [Wei88, Wei90, Wei97a, LW93, Wei97b, Wei94b, Wei94a, DSW98b, Wei99, Wei00, AW00]. The basic idea is to compute a finite set S of terms such that $\exists x.P(x)$ is equivalent to $\bigvee_{t \in S} P(t)$. The terms in S are sometimes called Skolem terms [Wei88, LW93] because they are the concrete witnesses in contrast to abstract Skolem functions. This technique was already used by Skolem but somehow forgotten until taken over by Cooper [Coo72] and later by others [RL78, FR75]. Weispfenning generalised this device to let S include *impure* terms which are not part of L, e.g. $+\infty$ of and ϵ representing an infinitesimal. These have to be properly handled by a *virtual substitution*, i.e. when "substituted" the result is again an L-formula but semantically equivalent to the substitution of an *intended* value, i.e. $\exists x.P(x) \leftrightarrow \bigvee_{t \in S} P(t//x)$. More generally virtual substitution of t can be restricted by assumptions θ_t (t can be impure). The qe. theorem has the form

$$\exists x.P(x) \leftrightarrow \bigvee_{t \in S} \theta_t \wedge P(t//x).$$

An example we encounter in §4.5 is substituting a "fraction" $t = \frac{a}{b}$ for x in P using the language of rings, but under the condition $\theta_t = (b \neq 0)$. This approach is often referred to as *generic* qe. and was introduced in [DSW98a]. Note that θ_t could "artificially" contain duplicate, or more precisely implicit, assumptions in order to optimise simplification in subsequent steps, cf. [DSW98a] for an ingenious use of this approach.

Definable sets and projections

Quantifier elimination has a geometrical characterisation. Consider an L-structure M and let ϕ be an L-formula with free variables $x_1, \ldots, x_n$, then ϕ defines a subset $\mathcal{S}(\phi) = \{(x_1, \ldots, x_n) \in M^n | M \models \phi(x_1, \ldots, x_n)\}$ of M^n. We say that $A \subseteq M^n$ is definable (resp. qf. definable) in M if there is an L-formula (resp. qf. formula) ϕ with n free variables, say $x_1, \ldots, x_n$, such that $A = \mathcal{S}(\phi)$. A function $f : M^n \to M^m$ is definable (resp. qf. definable) if its graph $\mathit{graph}(f) = \{(x_1, \ldots, x_n, y_1, \ldots, y_m) \in M^{n+m} | f(x_1, \ldots, x_n) = (y_1, \ldots, y_m)\}$ is definable (resp. qf. definable) in M. Clearly two formulae ϕ and ϕ' are equivalent if and only if $\mathcal{S}(\phi) = \mathcal{S}(\phi')$ and hence

Theorem 8. *M admits qe. if and only if every definable set $A \subseteq M^n$ in M is qf. definable in M.*

Since it is much easier to characterise the structure of qf. definable sets in M, qe. is a powerful tool to characterise the structure of all definable sets in M. Conversely if we can find a set $A \subseteq M^n$ structurally different from any qf. definable set in M, then M does not admit qe. For instance, Weispfenning used this observation to prove that the mixed real-integer theory of linear arithmetic does not admit qe. using a unary predicate true for $x \in \mathbb{R}$ if and only if $x \in \mathbb{Z}$. The study of definable sets is of great importance in model theory but also in several other fields of mathematics such as algebraic geometry where the main interest is the study of affine varieties mainly over $\mathbb{C}$ and $\mathbb{R}$. Also for formalisations in systems verification it is very useful to know that the formulae we use correspond to

$$
\begin{array}{lll}
(\!|\boldsymbol{T}|\!)^e_{i_\alpha} & = \mathit{True} & (\!|\neg p|\!)^e_{i_\alpha} = (\neg (\!|p|\!)^e_{i_\alpha}) \\
(\!|\boldsymbol{F}|\!)^e_{i_\alpha} & = \mathit{False} & (\!|p \wedge q|\!)^e_{i_\alpha} = ((\!|p|\!)^e_{i_\alpha} \wedge (\!|q|\!)^e_{i_\alpha}) \\
(\!|\boldsymbol{At}_a|\!)^e_{i_\alpha} & = i_\alpha\ e\ a & (\!|p \vee q|\!)^e_{i_\alpha} = ((\!|p|\!)^e_{i_\alpha} \vee (\!|q|\!)^e_{i_\alpha}) \\
(\!|\exists\, p|\!)^e_{i_\alpha} & = (\exists x.(\!|p|\!)^{x\cdot e}_{i_\alpha}) & (\!|p \longrightarrow q|\!)^e_{i_\alpha} = ((\!|p|\!)^e_{i_\alpha} \longrightarrow (\!|q|\!)^e_{i_\alpha}) \\
(\!|\forall\, p|\!)^e_{i_\alpha} & = (\forall x.(\!|p|\!)^{x\cdot e}_{i_\alpha}) & (\!|p \leftrightarrow q|\!)^e_{i_\alpha} = ((\!|p|\!)^e_{i_\alpha} \leftrightarrow (\!|q|\!)^e_{i_\alpha})
\end{array}
$$

Figure 4.1: Semantics of first order formulae

definable sets we want to describe. The study of definable sets seems to be one of the main motivations of Tarski's interest in qe., see some early papers [Tar31, KT31] by Tarski and Kuratowsky where they study definable sets of $\mathbb{R}$ based on Tarski's result for real closed fields (by then unpublished).

Quantifier elimination has a geometrical interpretation, using the notion of projection. Let $A \subseteq M^{n+1}$, for $n \geq 1$, then we define the projection of B by $\pi_{n+1}(A) = \{(x_1, \ldots, x_n) \in M^n | (x_1, \ldots, x_n, y) \in A \text{ for some } y \in M\}$.

Theorem 9. *Let M be an L-structure, then the following are equivalent:*

1. *M admits qe.*
2. *For every qf. definable $A \in M^{n+1}$, $\pi_{n+1}(A)$ is qf. definable.*
3. *For every qf. definable $A \subseteq M^n$ and definable $f : M^n \to M^m$, $f(A)$ is qf. definable.*

In algebraic geometry one strong property of qe. is that the projection of a representable set in one direction is again a representable set. The sets of interest are generally the affine varieties and ideals of polynomials over fields (mostly $\mathbb{C}$ and $\mathbb{R}$).

Reflecting generic quantifier elimination

In §2.1.3 we have presented generic qe. as a conversional. In the following, we present generic qe. based on reflection. This gives a preliminary idea on the common basis used by the reflected qe. procedures presented in this chapter. The qe. problem in its general form is not suited for a certificate-based integration. All procedures we consider in this chapter are integrated as derived rules or using reflection.

We use a datatype $\boldsymbol{\phi}_\alpha$ to reflect first-order formulae, whose atoms are of type α.

$$
\begin{array}{lll}
\mathsf{datatype}\ \boldsymbol{\phi}_\alpha & = & \boldsymbol{T} \mid \boldsymbol{F} \mid \boldsymbol{At}_\alpha \mid \neg\, \boldsymbol{\phi}_\alpha \mid \boldsymbol{\phi}_\alpha \wedge \boldsymbol{\phi}_\alpha \mid \boldsymbol{\phi}_\alpha \vee \boldsymbol{\phi}_\alpha \\
& & \mid \boldsymbol{\phi}_\alpha \longrightarrow \boldsymbol{\phi}_\alpha \mid \boldsymbol{\phi}_\alpha \leftrightarrow \boldsymbol{\phi}_\alpha \mid \exists\, \boldsymbol{\phi}_\alpha \mid \forall\, \boldsymbol{\phi}_\alpha
\end{array}
$$

A $\boldsymbol{\phi}_\alpha$-formula p must be understood in connection with its interpretation $(\!|p|\!)^e_{i_\alpha}$ in Figure 4.1. The interpretation of formulae depends on an environment $e :: [\beta]$ and on the interpretation of atoms $i_\alpha :: [\beta] \Rightarrow \alpha \Rightarrow \mathit{bool}$. The environment e is a list of *values* taken by the variables. The variables themselves are not part of the syntax of formulae, but only occur in atoms. This is the reason why e is only passed through recursively to reach i_α. In our applications, we will formalise variables using de Bruijn indices, which serve as an index in e for the interpretation. This is the reason why the quantifiers $\exists$ and $\forall$ carry no name. The interpretation of the quantifiers $\exists$ and $\forall$ inserts the bound variable at the head of e. Therefore, the variable with index 0 will be the bound variable in our applications.

Let $P :: \alpha \Rightarrow \mathit{bool}$ be a predicate on atoms and $p :: \boldsymbol{\phi}_\alpha$ be a qf. formula. Then we use $\mathsf{all}_{\boldsymbol{At}_\alpha}\ P\ p$ to express that all atoms in $p :: \boldsymbol{\phi}_\alpha$ satisfy P. Note that $\mathsf{all}_{\boldsymbol{At}_\alpha}\ (\lambda a.\mathit{True})$ describes exactly qf. formulae. Assume that $\mathsf{unbound}_\alpha :: \alpha \Rightarrow \mathit{bool}$ formalises the fact that an atom does not depend on the bound variable, then $\mathsf{all}_{\boldsymbol{At}_\alpha}\ \mathsf{unbound}_\alpha$ formalises that a given formula does not depend on the bound variable. Similarly to $\mathsf{all}_{\boldsymbol{At}_\alpha}$, we define $\mathsf{on}_{\boldsymbol{At}_\alpha} :: (\alpha \Rightarrow \alpha) \Rightarrow \boldsymbol{\phi}_\alpha \Rightarrow \boldsymbol{\phi}_\alpha$ to apply a function $f :: \alpha \Rightarrow \alpha$ on all atoms of a formula. For instance, if decr_α decrements all de Bruijn indices in atoms, then $\mathsf{on}_{\boldsymbol{At}_\alpha}\ \mathsf{decr}_\alpha$ decrements all indices in a qf. formula. Since most interesting operations on formulae depend on corresponding operations on atoms, we introduce the following locale:

$$\begin{array}{lcl}
\mathsf{qelim}\ qe\ (\forall\, p) & = & \neg qe(\mathsf{qelim}\ qe\ (\neg p)) \\
\mathsf{qelim}\ qe\ (\exists\, p) & = & qe(\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ (\neg p) & = & \neg(\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ (p \wedge q) & = & (\mathsf{qelim}\ qe\ p) \wedge (\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ (p \vee q) & = & (\mathsf{qelim}\ qe\ p) \vee (\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ (p \longrightarrow q) & = & (\mathsf{qelim}\ qe\ p) \longrightarrow (\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ (p \longleftrightarrow q) & = & (\mathsf{qelim}\ qe\ p) \longleftrightarrow (\mathsf{qelim}\ qe\ p) \\
\mathsf{qelim}\ qe\ p & = & \mathsf{simp}_{\boldsymbol{\phi}}\ p
\end{array}$$

Figure 4.2: Quantifier elimination for $\boldsymbol{\phi}_\alpha$-formulae

locale *atom* =
  fixes $i_\alpha :: [\beta] \Rightarrow \alpha \Rightarrow \mathit{bool}$ **and** $\mathsf{unbound}_\alpha :: \alpha \Rightarrow \mathit{bool}$
    and $\mathsf{decr}_\alpha :: \alpha \Rightarrow \alpha$ **and** $\mathsf{At} :: \alpha \Rightarrow \boldsymbol{\phi}_\alpha$ **and** $\mathsf{At}^\neg :: \alpha \Rightarrow \boldsymbol{\phi}_\alpha$
  assumes $\mathsf{unbound}_\alpha\ a \rightarrow (\forall x, y. i_\alpha\ (x \cdot e)\ a \leftrightarrow i_\alpha\ (y \cdot e)\ a)$
    and $\mathsf{unbound}_\alpha\ a \rightarrow (i_\alpha\ e\ (\mathsf{decr}_\alpha\ a) \leftrightarrow i_\alpha\ (x \cdot e)\ a)$
    and $(\!|\mathsf{At}\ a|\!)^e_{i_\alpha} \leftrightarrow (\!|\boldsymbol{At}_a|\!)^e_{i_\alpha}) \wedge ((\!|\mathsf{At}^\neg a|\!)^e_{i_\alpha} \leftrightarrow (\!|\neg(\boldsymbol{At}_a)|\!)^e_{i_\alpha})$
    and $\mathsf{qfree}\ (\mathsf{At}\ a) \wedge \mathsf{qfree}\ (\mathsf{At}^\neg a)$
    and $\mathsf{unbound}_\alpha\ a \rightarrow \mathsf{all}_{\boldsymbol{At}_\alpha}\ \mathsf{unbound}_\alpha\ (\mathsf{At}\ a) \wedge \mathsf{all}_{\boldsymbol{At}_\alpha}\ \mathsf{unbound}_\alpha\ (\mathsf{At}^\neg\ a)$

Functions $i_\alpha, \mathsf{unbound}_\alpha$ and decr_α are as described above, cf. their axioms. Furthermore, we require two functions At and $\mathsf{At}^\neg$ motivated by efficiency in practice. They are semantically equivalent to $\boldsymbol{At}$ and $\neg \circ \boldsymbol{At}$, respectively, but can be implemented more efficiently. If, for instance, α reflects relations on arithmetical terms, then $\mathsf{At}\ (\mathbf{0} = \mathbf{1})$ and $\mathsf{At}^\neg\ (\mathbf{0} = \mathbf{1})$ could return $\boldsymbol{F}$ and $\boldsymbol{T}$, respectively.

In our experience, using optimised versions of the constructors is invaluable. We define for every constructor $\boldsymbol{c}$ of $\boldsymbol{\phi}_\alpha$ a function $\underline{\boldsymbol{c}}$, which is semantically equivalent to $\boldsymbol{c}$. Here is for instance a simple optimised conjunction:

$$p \mathbin{\underline{\wedge}} q = \mathbf{if}\ p = \boldsymbol{F} \vee q = \boldsymbol{F}\ \mathbf{then}\ \boldsymbol{F}\ \mathbf{else\ if}\ p = \boldsymbol{T}\ \mathbf{then}\ q\ \mathbf{else\ if}\ q = \boldsymbol{T}\ \mathbf{then}\ p\ \mathbf{else}\ p \wedge q$$

We use the optimised constructors overall in this thesis. To avoid cumbersome notation, every occurrence of any constructor of $\boldsymbol{\phi}_\alpha$ in the rest of the thesis, except for pattern matching, represents the optimised version of that constructor. This also applies to any shadow syntax in the rest of the thesis. Based on optimised constructors, we define a canonical simplification $\mathsf{simp}_{\boldsymbol{\phi}}$ satisfying (4.1). Function $\mathsf{simp}_{\boldsymbol{\phi}}$ just simplifies sub-formulae and replaces any constructor by its optimised version.

$$\mathbf{(in}\ \mathit{atom}\mathbf{)} \quad (\!|\mathsf{simp}_{\boldsymbol{\phi}}\ p|\!)^e_{i_\alpha} \leftrightarrow (\!|p|\!)^e_{i_\alpha} \tag{4.1}$$

Assume that we have a function qe to eliminate on $\exists$ over a qf. body. Then we lift qe to a qep. for all $\boldsymbol{\phi}$-formulae using qelim in Figure 4.2. The main property (4.2) is straightforward.

$$\begin{array}{ll}
\mathbf{(in}\ \mathit{atom}\mathbf{)} & (\forall e, q.\ \mathsf{qfree}\ q \rightarrow \mathsf{qfree}\ (qe\ q) \wedge ((\!|qe\ q|\!)^e_{i_\alpha} \leftrightarrow (\!|\exists\, q|\!)^e_{i_\alpha})) \\
& \quad \rightarrow \mathsf{qfree}\ (\mathsf{qelim}\ qe\ p) \wedge ((\!|\mathsf{qelim}\ qe\ p|\!)^e_{i_\alpha} \leftrightarrow (\!|p|\!)^e_{i_\alpha}).
\end{array} \tag{4.2}$$

In practice, we must apply qe to formulae as small as possible, to overcome the strikingly bad complexity results for qe. and obtain relatively efficient qe. procedures. For that we must at least distribute $\exists$ over $\vee$ and eliminate conjunctions not depending on the bound variable, cf. (4.i) and (4.j). For this purpose, we introduce functions $\mathsf{split}_\wedge$ and $\mathsf{split}_\vee$ to return all conjuncts and disjuncts of a formula, respectively. Similarly, $\mathsf{list}_\wedge$ and $\mathsf{list}_\vee$ turn a list of formulae into their conjunction and disjunction, respectively. Given $f :: \boldsymbol{\phi}_\alpha \Rightarrow \boldsymbol{\phi}_\alpha$ and formulae $p_1, \ldots p_n$, then $\mathsf{eval}_\vee\ f\ [p_1, \ldots, p_n]$ returns $f\ p_1 \vee \ldots f\ p_n$ evaluated lazily. If f is a qep. for one $\exists$, then $\lambda p.\mathsf{eval}_\vee\ f\ (\mathsf{split}_\vee\ p)$ distributes f over disjunctions, and is therefore a

qep. for one $\exists$. Let $p :: \boldsymbol{\phi}_\alpha$ be qf. and $ps = \mathsf{split}_\wedge\ p$. Function $\mathsf{eval}_\wedge$ first partitions ps into two list: as consist of all formulae not involving the bound variable and bs consists of the rest. The call $\mathsf{eval}_\wedge\ f\ p$ then returns $\mathsf{on}_{\boldsymbol{At}_\alpha}\ \mathsf{decr}_\alpha\ (\mathsf{list}_\wedge\ as) \wedge f\ (\mathsf{list}_\wedge\ bs)$. If f is a qep. as above, then $\mathsf{eval}_\wedge\ f$ applies f only to the conjuncts not involving the bound variable. Summed up, given a qep. qe, then $\mathsf{eval}_\vee\ (\mathsf{eval}_\wedge\ qe)$ is an optimised qep. satisfying our purpose. In fact the real definition of qelim for $\exists$ is as follows:

$$\mathsf{qelim}\ qe\ (\exists\ p) = \mathsf{eval}_\vee\ (\mathsf{eval}_\wedge\ qe)\ p.$$

The main property (4.2) still holds for qelim changed as above. Further optimisation for using equations and Gaußelimination, i.e. $(\exists x.x = t \wedge P(x)) \leftrightarrow P(t)$ are also extremely useful in practice. See [Nip08] for an optimiser using equations. In §4.5 we use a quadratic equation to optimise qe. for real closed fields.

Unfortunately, I must disappoint the reader by pointing out that this generic treatment above is included for presentation reasons *only*. Its formalisation took 500 lines proofs in Isabelle. The reflected procedures in this chapter do not rely on the datatype $\boldsymbol{\phi}_\alpha$ but redefine it time and again. All presented functionalities are duplicated. It would be interesting to adapt all our developments in this generic framework. See [Nip08] for such a generic treatment for dense linear orders and linear arithmetic.

4.2 Dense linear orders, revisited

In contrast to §2.1.3 we formalise DLO by requiring a function which given any x and y returns an element $x \between y$ lying "between" them instead of the denseness axiom:

locale $dlo_\between = linorder\ +$ **fixes** $.\between.+$
assumes $\forall x.\exists y.y < x$
and $\forall x.\exists y.x < y$
and $\forall x, y.x < y \rightarrow x < x \between y < y$
and $\forall x.x \between x = x$

Note first that $dlo_\between$ indeed formalises DLO, i.e. it is an instance of dlo and hence we can already use our Langford's qe. of §2.1.3 for the sub-language not involving $\between$.

$$\textbf{instance}\ dlo_\between \subseteq dlo \tag{4.3}$$

This formalisation does not admit qe. considering $\between$ *as a language element.*

Theorem 10. *The theory of $dlo_\between$ does not admit qe. and is not complete.*

Proof. Consider the sentence $P := \forall x, y.x \between y = y \between x$. Assume for contradiction $dlo_\between$ has qe., then there is a qf. formula P' with no more free variables than P, such that $P \leftrightarrow P'$ is true in all models of $dlo_\between$. Since the only constants in the language of $dlo_\between$ are *True* and *False*, these are the only two possibilities for P'. Now consider two models of $dlo_\between$ as the real numbers with the usual ordering and $x \between y$ is defined to be $\frac{x+y}{2}$ in the first and $\frac{x+2\cdot y}{3}$ in the second model. Clearly these are two models of $dlo_\between$ but P only holds in the first. Incompleteness follows from the same argument. □

To obtain qe. we must not consider $\between$ as a language element but rather as an impure term which must vanish after virtual substitution. In the following we show that Ferrante and Rackoff's qe. theorem for linear real arithmetic holds in $dlo_\between$, and hence obtain under acceptable conditions a qe. procedure for a class of special dense linear orders including linear arithmetic over ordered fields.

4.2.1 Ferrante and Rackoff's algorithm

Consider $\exists x.P(x)$, where P is qf. Furthermore we assume P to be *x-normalised*, i.e. it consists only of $\wedge, \vee$ and atoms of type (A) $x = t$, (B) $x < t$, (C) $x > t$ or (D) those not

involving the existentially bound variable x. In (A)–(C), t does not depend on x. Any qf. formula over the language of *dlo* (not $dlo_{\between}$, i.e. not involving $\between$) can be transformed into an *x-normalised* formula by means of NNF and simple syntactical manipulations.

The algorithm constructs the set $\mathcal{U}_P = \{t | x = t, x < t \text{ or } x > t \text{ occurs in P}\}$ and two formulae $P_{-\infty}$ and $P_{+\infty}$ obtained from P by replacing atoms of type (A),(B) and (C) by *False*, *True*, and *False*, respectively, (for $P_{-\infty}$) and by *False*,*False*, and *True*, respectively, (for $P_{+\infty}$). The qe. is now a consequence of Theorem 11.

Theorem 11 (Lemma 1.1 in [FR75])**.** *An x-normalised P satisfies:*

$$(\exists x.P(x)) \leftrightarrow P_{-\infty} \vee P_{+\infty} \vee \exists (u, u') \in \mathcal{U}_P^2.P(u \between u').$$

Proof. We give an alternative proof to [FR75].

$\boldsymbol{P_{-\infty} \vee P_{+\infty} \vee \exists (u,u') \in \mathcal{U}_P^2.P(u \between u') \rightarrow \exists x.P(x)}$ Obviously $\exists (u, u') \in \mathcal{U}_P^2.P(u \between u') \rightarrow \exists x.P(x)$ holds. The cases $P_{-\infty} \rightarrow \exists x.P(x)$ and $P_{+\infty} \rightarrow \exists x.P(x)$ follow directly from (4.k) and (4.l), the main properties of $P_{-\infty}$ (resp. $P_{+\infty}$), proved by induction. They state that $P_{-\infty}$ (resp. $P_{+\infty}$) simulate the behaviour of P for arbitrarily small (resp. big) values.

$$\exists y. \forall x < y.P(x) \leftrightarrow P_{-\infty} \tag{4.k}$$

$$\exists y. \forall x > y.P(x) \leftrightarrow P_{+\infty} \tag{4.l}$$

$\boldsymbol{\neg P_{-\infty} \wedge \neg P_{+\infty} \wedge \exists x.P(x) \rightarrow \exists (u,u') \in \mathcal{U}_P^2.P(u \between u')}$ Assume $P(x)$ for some x and $\neg P_{-\infty}$ and $\neg P_{+\infty}$, i.e. x is a neither "too large" nor "too small" witness for P. First note that x must lie between two points in $\mathcal{U}_P$, a trivial consequence of (4.m) and (4.n), both proved by induction.

$$\forall x. \neg P_{-\infty} \wedge P(x) \rightarrow \exists l \in \mathcal{U}_P.l \leq x \tag{4.m}$$

$$\forall x. \neg P_{+\infty} \wedge P(x) \rightarrow \exists u \in \mathcal{U}_P.x \leq u. \tag{4.n}$$

Ultimately either $x \in \mathcal{U}_P$, in which case we are done since $x \between x = x$, or there is a *smallest* interval with endpoints in $\mathcal{U}_P$ containing x, i.e. $l_x < x < u_x \wedge \forall y. l_x < y < u_x \rightarrow y \notin \mathcal{U}_P$ for some $]l_x, u_x[\in \mathcal{U}_P^2$. The *smallest* interval construction is simple since $\mathcal{U}_P$ is finite. Finally, consider an arbitrary y and assume $y \in (l_x, u_x)$. Proving $P(y)$ finishes the proof: take $u = l_x$ and $u' = u_x$. This property shows the expressibility limitations of DLO: P does not change its truth value over smallest intervals with endpoints in $\mathcal{U}_P$, i.e.

$$\begin{aligned} \forall x, l, u. (\forall y. l < y < u \rightarrow y \notin \mathcal{U}_P) \wedge l < x < u \wedge P(x) \\ \rightarrow \forall y. l < y < u \rightarrow P(y) \end{aligned} \tag{4.o}$$

The proof of (4.o) is by induction on P. If P is of type (A) the result trivially holds. For the case $x < t$, fix an arbitrary y and assume $l < y < u$. Note that $y \neq t$ since $t \in \mathcal{U}_P$. Hence $y < t$, i.e. $P(y)$, for if $y > t$ then $l < t < u$, which contradicts the premises since $t \in \mathcal{U}_P$. The case $x > t$ is analogous and the $\wedge$ and $\vee$-cases are trivial. □

We present a context-aware and proof-producing implementation of this qep.

4.2.2 A derived rule

Theorem 11, revisited

The Isabelle proof of Theorem 11 only needs the finiteness of U, a HOL-set representing $\mathcal{U}_P$, and the properties (4.k)–(4.o), which correspond to the premises of (4.4).

$$\begin{aligned} \textbf{(in } dlo_{\between}\textbf{)} \ \mathsf{finite}\ U \wedge (\exists y. \forall x < y.P(x) \leftrightarrow Q) \wedge (\exists y. \forall x > y.P(x) \leftrightarrow R) \wedge \\ (\forall x. \neg Q \wedge P(x) \rightarrow \exists l \in U.l \leq x) \wedge (\forall x. \neg R \wedge P(x) \rightarrow \exists u \in U.x \leq u) \wedge \\ (\forall x, l, u. (\forall y. l < y < u \rightarrow y \notin U) \wedge l < x < u \wedge P(x) \rightarrow \forall y. l < y < u \rightarrow P(y)) \\ \rightarrow \exists x.P(x) \leftrightarrow (Q \vee R \vee \exists (u, u') \in U^2.P(u \between u')). \end{aligned} \tag{4.4}$$

The Isabelle/HOL proof of (4.4) follows exactly the proof of Theorem 11. Given a problem instance, we synthesise the premises of (4.4) using the techniques in §2.1.1 and hence obtain the qe. equivalence. For the proof synthesis of (4.k), we first automatically prove (4.5), (4.6) and (4.7). Let $(4.5)_\beta$ denote the respective theorem in (4.5) for every $\beta \in \{<, >, =\}$ and analogously for $(4.7)_\diamond$ and $\diamond \in \{\wedge, \vee\}$. The proof-synthesis of (4.k) for an x-normalised P is then the result of $prove_{-\infty}\ x\ P$.

$$\textbf{(in } dlo_{\emptyset}\textbf{)}: \ \exists z.\forall x < z.x < t \leftrightarrow \mathit{True} \quad \exists z.\forall x < z.x \bowtie t \leftrightarrow \mathit{False}, \text{ for } \bowtie \in \{>, =\} \tag{4.5}$$

$$\textbf{(in } dlo_{\emptyset}\textbf{)}\ \exists z.\forall x < z.F \leftrightarrow F \tag{4.6}$$

$$\textbf{(in } dlo_{\emptyset}\textbf{)}\ (\exists z.\forall x < z.P(x) \leftrightarrow P') \wedge (\exists z.\forall x < z.Q(x) \leftrightarrow Q') \rightarrow \exists z.\forall x < z.(P(x) \diamond Q(x)) \leftrightarrow (P' \diamond Q'), \text{ for } \diamond \in \{\wedge, \vee\}. \tag{4.7}$$

$decomp_{-\infty}\ x\ P =$
case P **of**
  $y\,\beta\,t | y = x | \beta \in \{<, >, =\} \Rightarrow ([], \boldsymbol{\lambda}[].(4.5)_\beta[t])$
  $A \diamond B | \diamond \in \{\wedge, \vee\} \Rightarrow ([A, B], \mathit{fwd}\ (4.7)_\diamond)$
  $\_ \Rightarrow ([], \boldsymbol{\lambda}[].(4.6)[P])$
$prove_{-\infty}\ x = \mathit{thm\text{-}of}\ (decomp_{-\infty}\ x)$

The proof-synthesis of (4.l)–(4.o) follows analogously. We exhibit proof-synthesis for (4.m), since it is representative for (4.m)–(4.o) and these depend on $\mathfrak{U}_P$. First, we need (4.8),(4.9) and (4.10). Let, as before, $(4.8)_\beta$ denote the respective theorem in (4.8) for every $\beta \in \{<, >, =\}$ and analogously for $(4.10)_\diamond$ and $\diamond \in \{\wedge, \vee\}$. For an x-normalised P, the function call $\mathit{uset}\ x\ P$ returns a function inU that for every $t \in \mathfrak{U}_P$ returns a theorem th **as** '$t \in U$', where U is a HOL set representing $\mathfrak{U}_P$. Technically this is realised by proving all these theorems once and for all and storing them in a table. Function inU then just performs look up. The synthesis of (4.m) is the result of $prove_{-\infty\mathfrak{U}}$.

$$\textbf{(in } dlo_{\emptyset}\textbf{)}\ t \in U \rightarrow (\forall x.\neg \mathit{True} \wedge x < t \rightarrow \exists l \in U.l \leq x)$$

$$t \in U \rightarrow (\forall x.\neg \mathit{False} \wedge x > t \rightarrow \exists l \in U.l \leq x)$$

$$t \in U \rightarrow (\forall x.\neg \mathit{False} \wedge x = t \rightarrow \exists l \in U.l \leq x) \tag{4.8}$$

$$\textbf{(in } dlo_{\emptyset}\textbf{)}\ \forall x.\neg F \wedge F \rightarrow \exists l \in U.l \leq x \tag{4.9}$$

$$\textbf{(in } dlo_{\emptyset}\textbf{)}\ (\forall x.\neg P' \wedge P(x) \rightarrow \exists l \in U.l \leq x) \wedge (\forall x.\neg Q' \wedge Q(x) \rightarrow \exists l \in U.l \leq x) \rightarrow \forall x.\neg(P' \diamond Q') \wedge (P(x) \diamond Q(x)) \rightarrow \exists l \in U.l \leq x, \text{ for } \diamond \in \{\wedge, \vee\}. \tag{4.10}$$

$decomp_{-\infty\mathfrak{U}}\ x\ inU\ P =$
case P **of**
  $y\,\beta\,t | y = x | \beta \in \{<, >, =\} \Rightarrow ([], \boldsymbol{\lambda}[].\mathit{fwd}\ (4.8)_\beta\ [inU\ t])$
  $A \diamond B | \diamond \in \{\wedge, \vee\} \Rightarrow ([A, B], \mathit{fwd}\ (4.10)_\diamond)$
  $\_ \Rightarrow ([], \boldsymbol{\lambda}[].(4.9)[P])$
$prove_{-\infty\mathfrak{U}}\ x\ P = \mathit{thm\text{-}of}\ (decomp_{-\infty\mathfrak{U}}\ x\ (\mathit{uset}\ x\ P))\ P$

The actual implementation synthesises (4.k)–(4.o) at once using $prove_{\pm\infty\mathfrak{U}}$, in order to avoid traversing the formula time and again. It returns a 6-elements list corresponding to the premises of (4.4).

The overall procedure

Assume we have a conversion *normalise*, which given a variable x and a qf. P returns $P = Q$, where Q is *x-normalised*. For an input $\exists x.P(x)$, $ferrack_\exists$ obtains a theorem $\exists x.P(x) \leftrightarrow \exists x.Q(x)$, where Q is x-normalised and synthesises the corresponding instance of Theorem 11. The call $\mathit{simplify}\ Q'$ returns th **as** '$Q' \leftrightarrow Q''$', where, among other simplifications, the bounded $\exists$ has been eliminated from Q'.

$ferrack_\exists\ \exists x.P(x) =$
let
  nth **as** '$\exists x.P(x) \leftrightarrow \exists x.Q(x)$' $= argc(absc(normalise\ x))\exists x.P(x)$
  $frth$ **as** '$\exists x.Q(x) \leftrightarrow Q'$' $= fwd\ (4.4)\ (prove_{\pm\mathfrak{U}}\ x\ Q)$
in $fwd\ trans\ [fwd\ trans\ [nth, frth], simplify\ Q']$

$ferrack = liftqe\ ferrack_\exists$

4.2.3 Integration as a context-sensitive method

The proof-method so far only relies on (4.4) and all the theorems needed to synthesise (4.k), (4.l), (4.m), (4.n) and (4.o) and on an ML function *normalise*. These constitute the two parts of the context-data, logical and non-logical respectively. We insert the logical data into the context-data via fact-declaration as usual, and provide a declaration to the user/developer to replace the default normalisation function (identity).

It is important to note here that the result of $ferrack_\exists$ is a theorem $(\exists x.P(x)) \leftrightarrow Q$, where Q involves $\between$. Hence, in order to work properly, the implementation above heavily relies on the fact that in the working instance of $dlo_\between$, the applications of $\between$ "disappears" after substitution or after simplification (i.e. *simplify*). In either case we say that $\between$ is definable in the structure instantiating $dlo_\between$ and is together with the existence of *normalise*, a function to *x-normalise* a qf. formula. The formalisation of the presented procedure took 500 lines of Isabelle proofs and 380 lines of SML code.

4.2.4 Linear arithmetic for ordered fields (almost) for free

Linear arithmetic over ordered fields with $x \between y = \frac{x+y}{2}$, is an instance of $dlo_\between$. To obtain qe. we only need to define *normalise*. But this is very simple since our implementation in §3.2 works for general polynomials, and hence for linear ones in particular: normalise terms in atoms, and transform them into one of the forms (A)-(D). In this way, we easily obtain a qep. for linear arithmetic over ordered fields in Isabelle/HOL. This adaptation took further 100 lines of Isabelle proofs and 200 lines of SML code.

With more care, we could obtain a qep. for the same theory above, but where the coefficients of the bound variables are multivariate polynomials in *another* set of variables. Normalisation works as above since the normaliser of §3.2 works for general polynomials. The transformation of $a \cdot x + b \bowtie 0$, for $\bowtie \in \{=, \neq, <\}$, a parameter polynomial a and b (not involving x) into one form (A)-(D), needs case splitting over the sign of a. The resulting procedure would be rather inefficient due to huge duplications. We investigate this theory separately in §4.3.1 using reflection and give a more efficient solution.

4.3 Linear arithmetics

In this section we present qe. procedures for three theories of linear arithmetic: parametric problems in ordered fields $\mathcal{F}_+^{\boldsymbol{\rho}}$ which generalises $\mathcal{R}_+$ (linear arithmetic over $\mathbb{R}$), Presburger arithmetic $\mathcal{Z}_+$ and the mixed real-integer arithmetic $\mathcal{R}_{\lfloor . \rfloor}$.

4.3.1 Parametric linear problems in ordered fields

We present a reflected qep. for $\mathcal{F}_+^{\boldsymbol{\rho}}$, the first order theory of parametric linear arithmetic over ordered fields. The only difference to $\mathcal{R}_+$, linear arithmetic over ordered fields, is that the coefficients of variables bound by quantifiers, are multivariate polynomials in a different set of variables, called parameters. Note that $\mathcal{F}_+^{\boldsymbol{\rho}}$ is more expressive than $\mathcal{R}_+$. Consider for instance $P(x) = a^2 = 2 \wedge x < a$. Then $\mathcal{S}(P) =]-\infty, \sqrt{2}[$, which is not expressible in $\mathcal{R}_+$. By the qe. result for $\mathcal{R}_+$, expressible sets in $\mathcal{R}_+$ are finite unions of intervals whose endpoints are infinite of *rational*. For $\mathcal{F}_+^{\boldsymbol{\rho}}$ expressible sets are exactly finite unions of intervals, whose endpoints are infinite of *algebraic*.

$$
\begin{array}{ll}
(\!|\widetilde{c}|\!)^e_\pi & = (\!|c|\!)^\pi_{\boldsymbol{\rho}} \\
(\!|\boldsymbol{u}_n|\!)^e_\pi & = e!n \\
(\!|-t|\!)^e_\pi & = -(\!|t|\!)^e_\pi \\
(\!|t+s|\!)^e_\pi & = (\!|t|\!)^e_\pi + (\!|s|\!)^e_\pi \\
(\!|t-s|\!)^e_\pi & = (\!|t|\!)^e_\pi - (\!|s|\!)^e_\pi \\
(\!|c*t|\!)^e_\pi & = (\!|c|\!)^\pi_{\boldsymbol{\rho}} \cdot (\!|t|\!)^e_\pi
\end{array}
\qquad
\begin{array}{ll}
(\!|\boldsymbol{T}|\!)^e_\pi & = \mathit{True} \\
(\!|\boldsymbol{F}|\!)^e_\pi & = \mathit{False} \\
(\!|t \bowtie s|\!)^e_\pi & = ((\!|t|\!)^e_\pi \bowtie (\!|s|\!)^e_\pi) \\
(\!|\neg p|\!)^e_\pi & = (\neg (\!|p|\!)^e_\pi) \\
(\!|p \diamond q|\!)^e_\pi & = ((\!|p|\!)^e_\pi \diamond (\!|q|\!)^e_\pi) \\
(\!|\exists\, p|\!)^e_\pi & = (\exists x.(\!|p|\!)^{x\cdot e}_\pi) \\
(\!|\forall\, p|\!)^e_\pi & = (\forall x.(\!|p|\!)^{x\cdot e}_\pi)
\end{array}
$$

Figure 4.3: Semantics of parametric linear arithmetic formulae

p	U_p	p_-	p_+
$q \diamond r$	$\mathsf{U}_q \mathbin{@} \mathsf{U}_r$	$q_- \diamond r_-$	$q_+ \diamond r_+$
$c * \boldsymbol{u}_0 + t = \boldsymbol{0}_{\boldsymbol{\rho}}$	$[(t,c)]$	$c = t = \boldsymbol{0}_{\boldsymbol{\rho}}$	$c = t = \boldsymbol{0}_{\boldsymbol{\rho}}$
$c * \boldsymbol{u}_0 + t \neq \boldsymbol{0}_{\boldsymbol{\rho}}$	$[(t,c)]$	$c = \boldsymbol{0}_{\boldsymbol{\rho}} \neq t \vee c \neq \boldsymbol{0}_{\boldsymbol{\rho}}$	$c = \boldsymbol{0}_{\boldsymbol{\rho}} \neq t \vee c \neq \boldsymbol{0}_{\boldsymbol{\rho}}$
$c * \boldsymbol{u}_0 + t < \boldsymbol{0}_{\boldsymbol{\rho}}$	$[(t,c)]$	$t < \boldsymbol{0}_{\boldsymbol{\rho}} = c \vee \boldsymbol{0}_{\boldsymbol{\rho}} < c$	$t < \boldsymbol{0}_{\boldsymbol{\rho}} = c \vee c < \boldsymbol{0}_{\boldsymbol{\rho}}$
$c * \boldsymbol{u}_0 + t \leq \boldsymbol{0}_{\boldsymbol{\rho}}$	$[(t,c)]$	$t \leq \boldsymbol{0}_{\boldsymbol{\rho}} = c \vee \boldsymbol{0}_{\boldsymbol{\rho}} \leq c$	$t \leq \boldsymbol{0}_{\boldsymbol{\rho}} = c \vee c \leq \boldsymbol{0}_{\boldsymbol{\rho}}$
-	$[]$	p	p

Figure 4.4: Definition of U_p, p_- and p_+

The syntax of terms and formulae is as follows:

$$
\begin{array}{ll}
\mathsf{datatype}\ \boldsymbol{\tau} & = \widetilde{\boldsymbol{\rho}} \mid \boldsymbol{u}_{\mathbb{N}} \mid -\boldsymbol{\tau} \mid \boldsymbol{\tau} + \boldsymbol{\tau} \mid \boldsymbol{\tau} - \boldsymbol{\tau} \mid \boldsymbol{\rho} * \boldsymbol{\tau} \\
\mathsf{datatype}\ \boldsymbol{\phi} & = \boldsymbol{T} \mid \boldsymbol{F} \mid \boldsymbol{\tau} = \boldsymbol{\tau} \mid \boldsymbol{\tau} \neq \boldsymbol{\tau} \mid \boldsymbol{\tau} < \boldsymbol{\tau} \mid \boldsymbol{\tau} \leq \boldsymbol{\tau} \\
& \quad \mid \neg\, \boldsymbol{\phi} \mid \boldsymbol{\phi} \wedge \boldsymbol{\phi} \mid \boldsymbol{\phi} \vee \boldsymbol{\phi} \mid \boldsymbol{\phi} \rightarrow \boldsymbol{\phi} \mid \boldsymbol{\phi} \leftrightarrow \boldsymbol{\phi} \mid \exists\, \boldsymbol{\phi} \mid \forall\, \boldsymbol{\phi}
\end{array}
$$

Note that in Figure 4.3, the coefficients of potentially bound variables are *multivariate polynomials over another set of variables, not involved in quantifiers*. Recall $\boldsymbol{\rho}$-polynomials in Figure 3.1 from §3.1.2. We modify Ferrante and Rackoff's algorithm to cope with polynomials as coefficients. In the following we write $\boldsymbol{0}_{\boldsymbol{\tau}}$ as as shorthand for $\widetilde{\boldsymbol{0}_{\boldsymbol{\rho}}}$. Let $\mathsf{unbound}_{\boldsymbol{\tau}}\ t$ (resp. $\mathsf{unbound}_{\boldsymbol{\phi}}\ p$) formalise that the $\boldsymbol{\tau}$-term t (resp. $\boldsymbol{\phi}$-formula p) does not contain $\boldsymbol{u}_0$ and $\mathsf{decr}_{\boldsymbol{\phi}}\ p$ be p with all de Brujin indices $\boldsymbol{u}_n$ decremented. These are simple recursive functions that satisfy:

$$\mathsf{unbound}_{\boldsymbol{\tau}}\ t \rightarrow \forall x, y.(\!|t|\!)^{x\cdot e}_\pi = (\!|t|\!)^{y\cdot e}_\pi \tag{4.11}$$

$$\mathsf{unbound}_{\boldsymbol{\phi}}\ p \rightarrow \forall x, y.(\!|p|\!)^{x\cdot e}_\pi \leftrightarrow (\!|p|\!)^{y\cdot e}_\pi \wedge \forall x.(\!|\mathsf{decr}_{\boldsymbol{\phi}}\ p|\!)^e_\pi \leftrightarrow (\!|p|\!)^{x\cdot e}_\pi \tag{4.12}$$

Let a $\boldsymbol{\phi}$-formula p be linear ($\mathsf{islin}_{\boldsymbol{\phi}}\ p$) if it does not involve $\boldsymbol{u}_0$ ($\mathsf{unbound}_{\boldsymbol{\phi}}\ p$) or has the form $f \diamond g$, for linear f and g and $\diamond \in \{\wedge, \vee\}$, or $c * \boldsymbol{u}_0 + r \bowtie \boldsymbol{0}_{\boldsymbol{\tau}}$, for $\bowtie \in \{<, \leq, =, \neq\}$, a normalised polynomial $c \neq \boldsymbol{0}_{\boldsymbol{\rho}}$ and $r :: \boldsymbol{\tau}$ not involving $\boldsymbol{u}_0$. Note that any qf. $\boldsymbol{\phi}$-formula is transformed into an equivalent linear $\boldsymbol{\phi}$-formula by $\mathsf{lin}_{\boldsymbol{\phi}}$. We do not present this in detail, but see [Cha06b, CN06]. The important property is

$$\mathsf{qfree}\ p \rightarrow \mathsf{islin}_{\boldsymbol{\phi}}\ (\mathsf{lin}_{\boldsymbol{\phi}}\ p) \wedge (\!|\mathsf{lin}_{\boldsymbol{\phi}}\ p|\!)^e_\pi \leftrightarrow (\!|p|\!)^e_\pi \tag{4.13}$$

Figure 4.4 defines p_-, p_+, and U_p for a linear $\boldsymbol{\phi}$-formula p. They are analogues of $P_{-\infty}$, $P_{+\infty}$ and $\mathfrak{U}_P$ in §4.2 for $P = \lambda x.(\!|p|\!)^{x\cdot e}_\pi$, but encode the implicit dependency on the polynomial parameters into the resulting formula by explicit case distinction. Note that we abuse notation and write $a \bowtie_1 b \bowtie_2 c$ for $a \bowtie_1 b \wedge b \bowtie_2 c$, for $\bowtie_i \in \{=, \neq, <, \leq\}$.

It is very easy to verify that p_- and p_+ do not depend on $\boldsymbol{u}_0$ and that they mimic p for values small (resp. large) enough in the underlying ordered field.

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow \mathsf{unbound}_{\boldsymbol{\phi}}\ p_- \wedge \mathsf{unbound}_{\boldsymbol{\phi}}\ p_+ \tag{4.14}$$

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow \exists z. \forall x < z.(\!|p_-|\!)^{x\cdot e}_\pi \leftrightarrow (\!|p|\!)^{x\cdot e}_\pi \tag{4.15}$$

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow \exists z. \forall x > z.(\!|p_+|\!)^{x\cdot e}_\pi \leftrightarrow (\!|p|\!)^{x\cdot e}_\pi \tag{4.16}$$

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow \forall (t,c) \in \{\!|\mathsf{U}_p|\!\}.\mathsf{unbound}_{\boldsymbol{\tau}}\ t \wedge \mathsf{ishorn}\ c \wedge c \neq \boldsymbol{0}_{\boldsymbol{\rho}} \tag{4.17}$$

p	$p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]$
$p \wedge q$	$p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}] \wedge q[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]$
$p \vee q$	$p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}] \vee q[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]$
$a * \boldsymbol{u}_0 + r = \mathbf{0}_{\boldsymbol{\tau}}$	$\widetilde{c} = s = r = \mathbf{0}_{\boldsymbol{\tau}} \vee$
	$\widetilde{c} \neq \mathbf{0}_{\boldsymbol{\tau}} = \widetilde{d} \wedge a * t = (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c) * r \vee$
	$\widetilde{d} \neq \mathbf{0}_{\boldsymbol{\tau}} = \widetilde{c} \wedge a * s = (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast d) * r \vee$
	$\widetilde{c \circledast d} \neq \mathbf{0}_{\boldsymbol{\tau}} \wedge a * (d * t + c * s) = (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c \circledast d) * r$
$a * \boldsymbol{u}_0 + r < \mathbf{0}_{\boldsymbol{\tau}}$	$\widetilde{c} = \widetilde{d} = r = \mathbf{0}_{\boldsymbol{\tau}} \vee$
	$\widetilde{d} < \mathbf{0}_{\boldsymbol{\tau}} = c \wedge a * s < (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast d) * r \vee$
	$c = \mathbf{0}_{\boldsymbol{\tau}} < \widetilde{d} \wedge (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast d) * r < a * s \vee$
	$\widetilde{c} < \mathbf{0}_{\boldsymbol{\tau}} = d \wedge a * t < (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c) * r \vee$
	$d = \mathbf{0}_{\boldsymbol{\tau}} < \widetilde{c} \wedge (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c) * r < a * t \vee$
	$\mathbf{0}_{\boldsymbol{\tau}} < \widetilde{c \circledast d} \wedge (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c \circledast d) * r < a * (d * t + c * s) \vee$
	$\widetilde{c \circledast d} < \mathbf{0}_{\boldsymbol{\tau}} \wedge a * (d * t + c * s) < (\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c \circledast d) * r$
$a * \boldsymbol{u}_0 + r \leq \mathbf{0}_{\boldsymbol{\tau}}$	$\ldots$
$a * \boldsymbol{u}_0 + r \neq \mathbf{0}_{\boldsymbol{\tau}}$	$\ldots$
p	p

Figure 4.5: Modified substitution in $\boldsymbol{\phi}$-formulae

A proof similar to §4.2 yields the reflection of Ferrante and Rackoff's theorem:

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow \exists x.(\!|p|\!)_{\pi}^{x\cdot e} \leftrightarrow (\!|p_- \vee p_+|\!)_{\pi}^{x\cdot e} \vee \exists((t,c),(s,d)) \in \{\!|\mathsf{U}_p|\!\}^2.(\!|p|\!)_{\pi}^{(\frac{(\!|t|\!)_{\pi}^{y\cdot e}}{-2\cdot(\!|c|\!)_{\boldsymbol{\rho}}^{\pi}} + \frac{(\!|s|\!)_{\pi}^{y\cdot e}}{-2\cdot(\!|d|\!)_{\boldsymbol{\rho}}^{\pi}})\cdot e} \tag{4.18}$$

Note that for a full implementation only a modified substitution $p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]$ of the "expression" $\frac{t}{-2\cdot c} + \frac{s}{-2\cdot d}$ for $\boldsymbol{u}_0$ in p satisfying (4.19) is missing. We use the same technique as for p_- and p_+ and encode all case splits on parameters into the result, see Figure 4.5. Recall that p_- and p_+ are modified substitutions of very large values.

$$\begin{aligned}&\mathsf{islin}_{\boldsymbol{\phi}}\ p \wedge \mathsf{unbound}_{\boldsymbol{\tau}}\ t \wedge \mathsf{unbound}_{\boldsymbol{\tau}}\ s \rightarrow \\ (\!|p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]|\!)_{\pi}^{x\cdot e} \leftrightarrow (\!|p|\!)_{\pi}^{(\frac{(\!|t|\!)_{\pi}^{x\cdot e}}{-2\cdot(\!|c|\!)_{\boldsymbol{\rho}}^{\pi}} + \frac{(\!|s|\!)_{\pi}^{x\cdot e}}{-2\cdot(\!|d|\!)_{\boldsymbol{\rho}}^{\pi}})\cdot e}&\ \wedge \mathsf{unbound}_{\boldsymbol{\phi}}\ p[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]\end{aligned} \tag{4.19}$$

The proof of (4.19) is only interesting for atoms and we show only the case of $(a * \boldsymbol{u}_0 + r < \mathbf{0}_{\boldsymbol{\tau}})[\frac{-t}{2\cdot c} + \frac{-s}{2\cdot d}]$. The $\leq$-case is analogous and the $=$ and $\neq$ are even simpler. For this, fix x and environments e and π. Clearly there are 9 disjoint cases depending on the strict sign of $(\!|c|\!)_{\boldsymbol{\rho}}^{\pi}$ and $(\!|d|\!)_{\boldsymbol{\rho}}^{\pi}$. These are exactly the cases encoded in Figure 4.5. Assume $(\!|c \circledast d|\!)_{\boldsymbol{\rho}}^{\pi} > 0$, then $(\!|c|\!)_{\boldsymbol{\rho}}^{\pi} \neq 0 \wedge (\!|d|\!)_{\boldsymbol{\rho}}^{\pi} \neq 0$ and hence $\frac{(\!|t|\!)_{\pi}^{x\cdot e}}{-2\cdot(\!|c|\!)_{\boldsymbol{\rho}}^{\pi}} + \frac{(\!|s|\!)_{\pi}^{x\cdot e}}{-2\cdot(\!|d|\!)_{\boldsymbol{\rho}}^{\pi}} = -\frac{(\!|d*t+c*s|\!)_{\pi}^{x\cdot}}{(\!|\widehat{\mathbf{2}_{\boldsymbol{\tau}}} \circledast c \circledast d|\!)_{\boldsymbol{\rho}}^{\pi}}$. The claim now follows using the property $\forall a,b,c.b > 0 \rightarrow \frac{a}{b} < c \leftrightarrow a < c \cdot b$ and simple algebraic manipulations. The other cases are similar.

Finally, we implement $\mathsf{fr}_{\exists}$ to eliminate one $\exists$, and fr the full qep. below. The function call $\mathsf{eval}_{\vee}\ f\ [x_1,..,x_n]$ returns the disjunction $f\ x_1 \vee \ldots \vee f\ x_n$ lazily evaluated. We prove the main qe. theorem in (4.24)

$$\begin{aligned}\mathsf{fr}_{\exists}\ q \quad &= \quad \textbf{let}\ p = \mathsf{lin}_{\boldsymbol{\phi}}\ q\ ;\ U = \mathsf{allpairs}(\mathsf{remdups}\ (\mathsf{U}_p)) \\ &\quad\ \ \textbf{in}\ \mathsf{decr}_{\boldsymbol{\phi}}(p_- \vee\ p_+ \vee \mathsf{eval}_{\vee}\ (\lambda((t,c),(s,d)).q[\tfrac{-t}{2\cdot c} + \tfrac{-s}{2\cdot d}])\ U) \\ \\ \mathsf{fr} \quad &= \quad \mathsf{qelim}\ \mathsf{fr}_{\exists}\end{aligned}$$

$$\mathsf{qfree}\ (\mathsf{fr}\ p) \wedge (\!|\mathsf{fr}\ p|\!)_{\pi}^{e} \leftrightarrow (\!|p|\!)_{\pi}^{e} \tag{4.20}$$

Drawbacks and a better solution

By inspecting the previous modified substitution in Figure 4.5, it is not hard to predict that the resulting procedure yields huge formulae and hence is not practicable even for simple examples. Our tests corroborate this prediction. Figure 4.5 shows *many* duplications of case splits. Of course for one atom there are no such duplications, but keeping in mind that we substitute the same "fraction", the conditions on the coefficients involved in that fraction must not be encoded *at the atoms level* but rather globally to avoid duplication. We present in the following an alternative substitution and procedure to achieve this goal.

Let us first reconsider the qe. theorem (4.18) and in particular the substitution of the fraction on the RHS. Let p be linear and let (t, c) and (s, d) be two elements of U_p. Furthermore fix environments e and π and let $P = \lambda x.(\!|p|\!)^{x\cdot e}_{\pi}$ and $\overline{t}, \overline{s}, \overline{c}$ and $\overline{d}$ denote $(\!|t|\!)^{y\cdot e}_{\pi}, (\!|s|\!)^{y\cdot e}_{\pi}, (\!|c|\!)^{\pi}_{\boldsymbol{\rho}}$ and $(\!|d|\!)^{\pi}_{\boldsymbol{\rho}}$ respectively. Our goal is to construct a $\boldsymbol{\phi}$-formula semantically equivalent to $P(\frac{\overline{t}}{-\underline{2}\cdot\overline{c}} + \frac{\overline{s}}{-\underline{2}\cdot\overline{d}})$ but without case splits on c and d at the atoms-level. For that consider all sign combinations of $\overline{c}$ and $\overline{d}$. If both are zero then we have $P(0)$. If exactly one is zero, say $\overline{d}$, then we have $P(\frac{\overline{t}}{-\underline{2}\cdot\overline{c}})$ and the whole disjunction (for only such cases) reduces to $\exists(t, c) \in \{\!\{\mathsf{U}_p\}\!\}.\overline{c} \neq 0 \wedge P(\frac{\overline{t}}{-\underline{2}\cdot\overline{c}})$. For the last case we have $\overline{c} \neq 0 \neq \overline{d}$ and hence $\frac{\overline{t}}{-\underline{2}\cdot\overline{c}} + \frac{\overline{s}}{-\underline{2}\cdot\overline{d}} = \frac{\overline{d}\cdot\overline{t}+\overline{c}\cdot\overline{s}}{-\underline{2}\cdot\overline{c}\cdot\overline{d}}$. By considering this last fraction only two case splits on the strict sign of the denominator are necessary to obtain a simpler substitution. Summed up, we prove the following qe. theorem, where $p[a]_{\boldsymbol{\phi}}$ denotes the "normal" substitution of term a for $\boldsymbol{u}_0$ in p:

$$\begin{aligned}\mathsf{islin}_{\boldsymbol{\phi}}\ p \rightarrow (\!|\exists\, p|\!)^{e}_{\pi} \leftrightarrow (\!|p_- \vee p_+ \vee p[\mathbf{0}_{\boldsymbol{\tau}}]_{\boldsymbol{\phi}}|\!)^{y\cdot e}_{\pi} \vee \exists(t, c) \in \{\!\{\mathsf{U}_p\}\!\}.(\!|c|\!)^{\pi}_{\boldsymbol{\rho}} \neq 0 \wedge (\!|p|\!)_{\pi}^{\frac{(\!|t|\!)^{y\cdot e}_{\pi}}{-\underline{2}\cdot(\!|c|\!)^{\pi}_{\boldsymbol{\rho}}}\cdot e} \vee \\ \exists((t, c), (s, d)) \in \{\!\{\mathsf{U}_p\}\!\}^2.(\!|c|\!)^{\pi}_{\boldsymbol{\rho}} \neq 0 \wedge (\!|d|\!)^{\pi}_{\boldsymbol{\rho}} \neq 0 \wedge (\!|p|\!)_{\pi}^{\frac{(\!|d*t+c*s|\!)^{y\cdot e}_{\pi}}{-\underline{2}\cdot(\!|c*d|\!)^{\pi}_{\boldsymbol{\rho}}}\cdot e} \qquad (4.21)\end{aligned}$$

Note that we only need to find a substitution $p[\frac{t}{c}]^{\neq}_{\boldsymbol{\phi}}$ of a "fraction" $\frac{t}{c}$ with "non-zero" denominator. Our substitution first splits over the strict sign of c and then performs two modified substitutions of $\frac{t}{c}$: the first $p[\frac{t}{c}]^{>}_{\boldsymbol{\phi}}$ assumes $\overline{c} > 0$, and the second $p[\frac{t}{c}]^{<}_{\boldsymbol{\phi}}$ assumes $\overline{c} < 0$. The definition of $p[\frac{t}{c}]^{\neq}_{\boldsymbol{\phi}}$ and that of $p[\frac{t}{c}]^{>}_{\boldsymbol{\phi}}$ for atoms are simple:

$$p[\frac{t}{c}]^{\neq}_{\boldsymbol{\phi}} = c < \mathbf{0}_{\boldsymbol{\tau}} \wedge p[\frac{t}{c}]^{<}_{\boldsymbol{\phi}} \vee c > \mathbf{0}_{\boldsymbol{\tau}} \wedge p[\frac{t}{c}]^{>}_{\boldsymbol{\phi}}$$

$$(a * \boldsymbol{u}_0 + b \bowtie \mathbf{0}_{\boldsymbol{\tau}})[\frac{t}{c}]^{>}_{\boldsymbol{\phi}} = a * t + c * b \bowtie \mathbf{0}_{\boldsymbol{\tau}} \quad \text{for } \bowtie \in \{=, <, \leq\}$$

The definition of $p[\frac{t}{c}]^{<}_{\boldsymbol{\phi}}$ is analogous. It is not hard to prove:

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \wedge (\!|c|\!)^{\pi}_{\boldsymbol{\rho}} \bowtie 0 \rightarrow (\!|p[\frac{t}{c}]^{\bowtie}_{\boldsymbol{\phi}}|\!)^{x\cdot e}_{\pi} \leftrightarrow (\!|p|\!)_{\pi}^{\frac{(\!|t|\!)^{x\cdot e}_{\pi}}{(\!|c|\!)^{\pi}_{\boldsymbol{\rho}}}\cdot e} \quad \text{for } \bowtie \in \{>, <, \neq\}. \qquad (4.22)$$

Now we can implement a new version of $\mathsf{fr}_{\exists}$ by:

$$\begin{aligned}\mathsf{fr}_{\exists}\ q \quad = \quad & \mathbf{let}\ p = \mathsf{lin}_{\boldsymbol{\phi}}\ q\ ; U = \mathsf{remdups}\ (\mathsf{U}_p);\ U_2 = \mathsf{allpairs}\ U \\ & \mathbf{in}\ \mathsf{decr}_{\boldsymbol{\phi}}(p_- \vee\ p_+ \vee p[\mathbf{0}_{\boldsymbol{\tau}}]_{\boldsymbol{\phi}} \vee \mathsf{eval}_{\vee}\ (\lambda(t, c).q[\tfrac{t}{-\mathbf{2}_{\boldsymbol{\rho}} \circledast c}]_{\boldsymbol{\phi}})\ U \\ & \qquad \vee \mathsf{eval}_{\vee}\ (\lambda((t, c), (s, d)).q[\tfrac{d*t+c*s}{-\mathbf{2}_{\boldsymbol{\rho}} \circledast c \circledast d}]_{\boldsymbol{\phi}})\ U_2)\end{aligned}$$

$$\mathsf{fr} \quad = \quad \mathsf{qelim}\ \mathsf{fr}_{\exists}$$

Using the previous theorems we prove their correctness:

$$\mathsf{qfree}\ q \rightarrow \mathsf{qfree}\ (\mathsf{fr}_{\exists}\ q) \wedge ((\!|\mathsf{fr}_{\exists}\ q|\!)^{e}_{\pi} \leftrightarrow (\!|\exists\, q|\!)^{e}_{\pi}) \qquad (4.23)$$

$$\mathsf{qfree}\ (\mathsf{fr}\ p) \wedge ((\!|\mathsf{fr}\ p|\!)^{e}_{\pi} \leftrightarrow (\!|p|\!)^{e}_{\pi}) \qquad (4.24)$$

The full reflection of $\mathcal{F}^{\boldsymbol{\rho}}_{+}$ took 3000 lines of Isabelle proof, but counting the first bad approach (600 lines). We needed further 140 lines of SML code for reification and a tactic.

$$
\begin{array}{ll}
(\!|\widehat{i}|\!)^e_\iota & = i \\
(\!|\boldsymbol{v}_n|\!)^e_\iota & = e!n \\
(\!|-t|\!)^e_\iota & = -(\!|t|\!)^e_\iota \\
(\!|t+s|\!)^e_\iota & = (\!|t|\!)^e_\iota + (\!|s|\!)^e_\iota \\
(\!|t-s|\!)^e_\iota & = (\!|t|\!)^e_\iota - (\!|s|\!)^e_\iota \\
(\!|i*t|\!)^e_\iota & = i \cdot (\!|t|\!)^e_\iota
\end{array}
\qquad
\begin{array}{ll}
(\!|\boldsymbol{T}|\!)^e & = \mathit{True} \\
(\!|\boldsymbol{F}|\!)^e & = \mathit{False} \\
(\!|s<t|\!)^e & = ((\!|s|\!)^e_\iota < (\!|t|\!)^e_\iota) \\
(\!|s\leq t|\!)^e & = ((\!|s|\!)^e_\iota \leq (\!|t|\!)^e_\iota) \\
(\!|s=t|\!)^e & = ((\!|s|\!)^e_\iota = (\!|t|\!)^e_\iota) \\
(\!|s\neq t|\!)^e & = ((\!|s|\!)^e_\iota \neq (\!|t|\!)^e_\iota) \\
(\!|i \mid t|\!)^e & = (i \mid (\!|t|\!)^e_\iota) \\
(\!|i \nmid t|\!)^e & = (i \nmid (\!|t|\!)^e_\iota)
\end{array}
\qquad
\begin{array}{ll}
(\!|\neg p|\!)^e & = (\neg (\!|p|\!)^e) \\
(\!|p\wedge q|\!)^e & = ((\!|p|\!)^e \wedge (\!|q|\!)^e) \\
(\!|p\vee q|\!)^e & = ((\!|p|\!)^e \vee (\!|q|\!)^e) \\
(\!|p\longrightarrow q|\!)^e & = ((\!|p|\!)^e \longrightarrow (\!|q|\!)^e) \\
(\!|p=q|\!)^e & = ((\!|p|\!)^e \leftrightarrow (\!|q|\!)^e) \\
(\!|\exists\, p|\!)^e & = (\exists x.(\!|p|\!)^{x\cdot e}) \\
(\!|\forall\, p|\!)^e & = (\forall x.(\!|p|\!)^{x\cdot e})
\end{array}
$$

Figure 4.6: Semantics of $\mathcal{Z}_+$-terms and formulae

p	B_p	A_p	p_-	p_+	δ_p
$q \diamond r$	B_q @ B_r	A_q @ A_r	$q_- \diamond r_-$	$q_+ \diamond r_+$	$lcm\ \delta_q\ \delta_r$
$\boldsymbol{v}_0 < t$	$[]$	$[t]$	$\boldsymbol{T}$	$\boldsymbol{F}$	1
$\boldsymbol{v}_0 \leq t$	$[]$	$[t+\widehat{1}]$	$\boldsymbol{T}$	$\boldsymbol{F}$	1
$t < \boldsymbol{v}_0$	$[t]$	$[]$	$\boldsymbol{F}$	$\boldsymbol{T}$	1
$t \leq \boldsymbol{v}_0$	$[t-\widehat{1}]$	$[]$	$\boldsymbol{F}$	$\boldsymbol{T}$	1
$\boldsymbol{v}_0 = t$	$[t-\widehat{1}]$	$[t+\widehat{1}]$	$\boldsymbol{F}$	$\boldsymbol{F}$	1
$\boldsymbol{v}_0 \neq t$	$[t]$	$[t]$	$\boldsymbol{T}$	$\boldsymbol{T}$	1
$d \mid \boldsymbol{v}_0 + t$	$[]$	$[]$	p	p	d
$d \nmid \boldsymbol{v}_0 + t$	$[]$	$[]$	p	p	d
-	$[]$	$[]$	p	p	1

Figure 4.7: Definition of $\mathsf{A}_p, \mathsf{B}_p, p_-, p_+$ and δ_p

4.3.2 Presburger arithmetic

In this section we present a formally verified implementation of Cooper's qep. for Presburger arithmetic $\mathcal{Z}_+ = \mathrm{Th}(\mathbb{Z}, <, +, 0, 1)$. It is not hard to prove that $\mathcal{Z}_+$ in this form does not admit qe. and that we must add divisibility predicates $\lambda x.i \mid x$ for every constant $i :: \mathbb{Z}$, but see [Sko30] for an interesting alternative.

We formalise the syntax as follows and the semantics in Figure 4.6:

$$
\begin{array}{ll}
\mathsf{datatype}\ \iota & = \widehat{\mathbb{Z}} \mid \boldsymbol{v}_{\mathbb{N}} \mid -\iota \mid \iota + \iota \mid \iota - \iota \mid \mathbb{Z} * \iota \\
\mathsf{datatype}\ \phi & = \iota < \iota \mid \iota \leq \iota \mid \iota = \iota \mid \iota \neq \iota \mid \mathbb{Z} \mid \iota \mid \mathbb{Z} \nmid \iota \mid \boldsymbol{T} \mid \boldsymbol{F} \\
& \quad \mid \neg\, \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \longrightarrow \phi \mid \phi \leftrightarrow \phi \mid \exists\, \phi \mid \forall\, \phi
\end{array}
$$

We only present cooper, a function that eliminates one $\exists$ over a qf. formula. Let a ϕ-formula p be linear ($\mathsf{islin}_\phi\ p$) if it does not involve $\boldsymbol{v}_0$ (i.e. $\mathsf{unbound}_\phi\ p$), has the form $f \diamond g$, where $\diamond \in \{\wedge, \vee\}$ and f and g are linear, or $\boldsymbol{v}_0 \bowtie_0 t$, $t \bowtie_1 \boldsymbol{v}_0$ or $d \bowtie_2 \boldsymbol{v}_0 + t$, where $\bowtie_0 \in \{<, \leq, =, \neq\}, \bowtie_1 \in \{<, \leq\}, \bowtie_2 \in \{|, \nmid\}, d > 0$ and $t :: \iota$ not involving $\boldsymbol{v}_0$ (i.e. $\mathsf{unbound}_\iota\ t$). Note that any qf. ϕ-formula is transformed into an equivalent linear one by lin_ϕ. This is less trivial than in 4.3.1 and is done as follows:

1. perform NNF and gather the coefficients of $\boldsymbol{v}_0$ in atoms.
2. for the resulting formula p, compute $l = lcm\{c \mid c \cdot \boldsymbol{v}_0 \text{ occurs in } p\}$
3. multiply the relation sides of every atom containing $c \cdot \boldsymbol{v}_0$ by $\frac{l}{c}$
4. replace the resulting formula according to $\exists x.P(l \cdot x) \leftrightarrow \exists x.l \mid x \wedge P(x)$
5. the rest follows by simple arithmetical transformations (e.g. $d \mid t \leftrightarrow |d| \mid t$ or $-x + t < 0 \leftrightarrow x > t$ etc.)

See [CN05, CN06] for a detailed formalisation of this step. The important property is

$$\mathsf{qfree}\ p \longrightarrow \mathsf{islin}_\phi\ (\mathsf{lin}_\phi\ p) \wedge (\!|\mathsf{lin}_\phi\ p|\!)^e \leftrightarrow (\!|p|\!)^e \tag{4.25}$$

For a linear formula p Cooper's theorems (4.26) and (4.27) give a qf. equivalent to $\exists\, p$,

based on functions $\mathsf{A}_p, \mathsf{B}_p, p_-, p_+$ and δ_p defined over the syntax of p in Figure 4.7.

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \to (\!|\exists\, p|\!)^e \leftrightarrow \exists j \in \{1..\delta_p\}.(\!|p_-|\!)^{j\cdot e} \vee \exists j \in \{1..\delta_p\}.\exists b \in \{\!\{\mathsf{B}_p\}\!\}.(\!|p|\!)^{((\!|b|\!)_\iota^{y\cdot e}+j)\cdot e} \tag{4.26}$$

$$\mathsf{islin}_{\boldsymbol{\phi}}\ p \to (\!|\exists\, p|\!)^e \leftrightarrow \exists j \in \{1..\delta_p\}.(\!|p_+|\!)^{j\cdot e} \vee \exists j \in \{1..\delta_p\}.\exists a \in \{\!\{\mathsf{A}_p\}\!\}.(\!|p|\!)^{((\!|a|\!)_\iota^{y\cdot e}-j)\cdot e} \tag{4.27}$$

Proof. For the proof of (4.26) fix e, assume $\mathsf{islin}_{\boldsymbol{\phi}}\ p$ and let $P = \lambda x.(\!|p|\!)^{x\cdot e}$, $P_\infty = \lambda x.(\!|p_-|\!)^{x\cdot e}$, $\delta = \delta_p$ and $\mathcal{B} = \{(\!|b|\!)_\iota^{y\cdot e} \mid b \in \{\!\{\mathsf{B}_p\}\!\}\}$. Clearly if $\exists j \in \{1..\delta\}, b \in \mathcal{B}.P(b+j)$ then $\exists x.P(x)$. It hence remains to prove $\exists j \in \{1..\delta\}.P_{-\infty}(j) \to \exists x.P(x)$ and $(\exists x.P(x)) \wedge \neg(\exists j \in \{1..\delta\}, b \in \mathcal{B}.P(b+j)) \to \exists j \in \{1..\delta\}.P_{-\infty}(j)$.

1. $\exists j \in \{1..\delta\}.P_{-\infty}(j) \to \exists x.P(x)$

To prove this, we need the following properties of $P_{-\infty}$:

$$\exists z.\forall x < z.P(x) \leftrightarrow P_{-\infty}(x) \tag{4.28}$$

$$\forall x, k.P_{-\infty}(x) \leftrightarrow P_{-\infty}(x - k\cdot\delta). \tag{4.29}$$

These properties are proved by induction on p: (4.28) states that P and $P_{-\infty}$ coincide over arguments that are small enough ; (4.29) states that $P_{-\infty}(x)$ is unaffected by the subtraction of any number of multiples of δ, i.e. $\{x \mid P_{-\infty}(x)\}$ is a periodic set. Note that only divisibility relations $d \mid x + r$ occur in $P_{-\infty}$, where $d \mid \delta$ (cf. Figure 4.7).

Now assume that $P_{-\infty}(j)$ holds for some $1 \le j \le \delta$, then using (4.29) we can subtract enough multiples of δ to reach a number below the z from (4.28), and thus obtain a witness for P.

2. $(\exists x.P(x)) \wedge \neg(\exists j \in \{1..\delta\}, b \in \mathcal{B}.P(b+j)) \to \exists j \in \{1..\delta\}.P_{-\infty}(j)$

By the argument above of decreasing witnesses by multiples of δ, it is sufficient to prove

$$\forall x.\neg(\exists j \in \{1..\delta\}, b \in \mathcal{B}.x = b + j) \to P(x) \to P(x - \delta). \tag{4.30}$$

The proof of (4.30) is by induction on p. The cases $\wedge$ and $\vee$ are trivial. In the case $\boldsymbol{v}_0 = t$ we derive a contradiction by taking $j = 1$, since $t - \widehat{1} \in \mathsf{B}_p$. In the cases $\boldsymbol{v}_0 < t$ and $d \mid \boldsymbol{v}_0 + t$ the claim is immediate since $\delta > 0$ and $d \mid \delta$, respectively. For the case $t < \boldsymbol{v}_0$, assume that $(\!|t|\!)_\iota^{y\cdot e} + \delta \ge x$, hence $x = (\!|t|\!)_\iota^{y\cdot e} + j$ for some $1 \le j \le \delta$, which contradicts the assumption since $t \in \{\!\{\mathsf{B}_p\}\!\}$. The other cases are analogous. □

A duality principle

Function mirror below formalises the duality principle (p_- and B_p vs. p_+ and A_p) pointed out by Cooper in [Coo72]. The idea behind mirror is simple: if p reflects $P(x)$ then mirror p reflects $P(-x)$. We prove this and other properties in (4.31). In the following mirror helps optimising cooper.

$$\begin{array}{lcll}
\mathsf{mirror}\ (p \diamond q) & = & (\mathsf{mirror}\ p) \diamond (\mathsf{mirror}\ q) & \text{for } \diamond \in \{\wedge, \vee\} \\
\mathsf{mirror}\ (\boldsymbol{v}_0 \bowtie t) & = & \boldsymbol{v}_0 \bowtie (-t) & \text{for } \bowtie \in \{=, \neq\} \\
\mathsf{mirror}\ (\boldsymbol{v}_0 \bowtie t) & = & (-t) \bowtie \boldsymbol{v}_0 & \text{for } \bowtie \in \{<, \le\} \\
\mathsf{mirror}\ (t \bowtie \boldsymbol{v}_0) & = & \boldsymbol{v}_0 \bowtie (-t) & \text{for } \bowtie \in \{<, \le\} \\
\mathsf{mirror}\ (d \bowtie \boldsymbol{v}_0 + r) & = & d \bowtie \boldsymbol{v}_0 + (-r) & \text{for } \bowtie \in \{|, \nmid\} \\
\mathsf{mirror}\ p & = & p &
\end{array}$$

$$\begin{array}{l}
\mathsf{islin}_{\boldsymbol{\phi}}\ p \to \mathsf{islin}_{\boldsymbol{\phi}}\ (\mathsf{mirror}\ p) \wedge (\!|\mathsf{mirror}\ p|\!)^{-i\cdot e} \leftrightarrow (\!|p|\!)^{i\cdot e} \\
\qquad \wedge \forall e.\{(\!|t|\!)_\iota^e \mid t \in \{\!\{\mathsf{A}_p\}\!\}\} = \{-(\!|t|\!)_\iota^e \mid t \in \{\!\{\mathsf{B}_{(\mathsf{mirror}\ p)}\}\!\}\}
\end{array} \tag{4.31}$$

An implementation

The first step in the implementation of cooper is to normalise the formula and to choose the smaller elimination set (A_p or B_p)

$$\begin{aligned}\mathsf{choose}\ p = \ &\mathbf{let}\ q = \mathsf{lin}_{\phi}\ p;\ (A, B) = (\mathsf{remdups}\ (\mathsf{A}_q), \mathsf{remdups}\ (\mathsf{B}_q))\\ &\mathbf{in\ if}\ |B| \leq |A|\ \mathbf{then}\ (q, B)\ \mathbf{else}\ (\mathsf{mirror}\ q, A)\end{aligned}$$

The main property of choose is that it reduces the $+\infty$ case to the $-\infty$ case:

$$\mathsf{qfree}\ p \wedge \mathsf{choose}\ p = (q, S) \rightarrow \mathsf{islin}_{\phi}\ q \wedge (\!|\exists\ p|\!)^e \leftrightarrow (\!|\exists\, q|\!)^e \wedge \{\!\{S\}\!\} = \{\!\{\mathsf{B}_q\}\!\} \tag{4.32}$$

For (q, S), we generate the right-hand side of (4.26) and expand the bounded quantifiers into disjunctions:

$$\mathsf{exp}_{\vee}(q, S) \quad = \quad \mathsf{eval}_{\vee}\ (\lambda i.q_{-}[\widehat{i}])\ [1..\delta_q] \vee \mathsf{eval}_{\vee}\ (\lambda t.q[t])\ [t + \widehat{i} \leftarrow t \in S, i \in [1..\delta_q]]$$

The substitution of a term t for $\boldsymbol{v}_0$ in a formula p is performed by $p[t]$. Substitution also simplifies the formula: it evaluates ground terms and relations and performs some logical simplification. Finally we decrease the de Bruijn indices of the remaining variables using function decr_{ϕ}. The composition of decr_{ϕ} and $\mathsf{exp}_{\vee}$ preserves the interpretation:

$$\mathsf{islin}_{\phi}\ p \wedge \{\!\{S\}\!\} = \{\!\{\mathsf{B}_p\}\!\} \rightarrow (\!|\exists\, p|\!)^e \leftrightarrow (\!|\mathsf{decr}_{\phi}(\mathsf{exp}_{\vee}\ (p, B))|\!)^e \wedge \mathsf{qfree}(\mathsf{decr}_{\phi}(\mathsf{exp}_{\vee}\ (p, B))). \tag{4.33}$$

Finally cooper and $\mathsf{qe}_{\mathcal{Z}_+}$ are simple and so are the proofs of (4.34) and (4.35).

$$\mathsf{cooper} = \mathsf{decr} \circ \mathsf{exp}_{\vee} \circ \mathsf{choose}$$

$$\mathsf{qe}_{\mathcal{Z}_+} = \mathsf{qelim}\ \mathsf{cooper}$$

$$\mathsf{qfree}\ q \rightarrow \mathsf{qfree}\ (\mathsf{cooper}\ q) \wedge (\!|\mathsf{cooper}\ q|\!)^e \leftrightarrow (\!|\exists\, q|\!)^e \tag{4.34}$$

$$\mathsf{qfree}\ (\mathsf{qe}_{\mathcal{Z}_+}\ p) \wedge (\!|\mathsf{qe}_{\mathcal{Z}_+}\ p|\!)^e \leftrightarrow (\!|p|\!)^e \tag{4.35}$$

For the formalisation of $\mathcal{Z}_+$, we needed 2000 lines of Isabelle proofs.

Definable sets

Definable sets in $\mathcal{Z}_+$ are exactly the almost periodic sets. Let $A \subseteq \mathbb{Z}$ be periodic if there exists (a period) $k \in \mathbb{N}$ such that $k \geq 1$ and for any $x \in \mathbb{Z}$ we have $x \in A$ if and only if $x + k \in A$. $A \subseteq \mathbb{Z}$ is almost periodic if there exists $n \in \mathbb{Z}$ and periodic sets B and C such that $n \geq 1$, $A \cap]-\infty, -n] = B \cap]-\infty, -n]$ and $A \cap [n, +\infty[= C \cap [n, +\infty[$. A definable set hence differs from a periodic set only by a finite set, as suggested by (4.26) and (4.27): p_{-} and p_{+} are periodic and A_p and B_p are finite.

4.3.3 Mixed real-integer arithmetic

> Man versucht es zunächst mit einem Prädikat, denn das verursacht oft weniger Probleme. Wenn es nicht glingt, so hat man meistens den Keim eines negativen Resultats.
>
> *(V. Weispfenning)*

Weispfenning [Wei99] proved $\mathcal{R}_{\lfloor . \rfloor} = \mathrm{Th}(\mathbb{R}, <, +, \lfloor . \rfloor, 0, 1)$ to admit qe. by reducing the elimination of one $\exists$ to the elimination of two $\exists$: one over $\mathcal{Z}_+$ and one over $\mathcal{R}_+$. We only exhibit a formalisation of this reduction, but see §4.3.2, §4.3.1 and [Cha06b] for a formalisation of the full procedure. According to Weispfenning [Wei06] our formalisation is the first implementation of this qep., a special joy for advocates of theorem proving.

Note that the natural mixture of $\mathcal{R}_+$ and $\mathcal{Z}_+$ using a unary predicate $x \in \underline{\mathbb{Z}} \leftrightarrow \exists i.x = \underline{i}$, although decidable [BJW05], does not admit qe. (see [Wei99] for a very elegant proof). This

$$
\begin{array}{lcl}
(\!|\widehat{i}|\!)^e_\mu & = & \underline{i} \\
(\!|\boldsymbol{x}_n|\!)^e_\mu & = & e!n \\
(\!|-t|\!)^e_\mu & = & -(\!|t|\!)^e_\mu \\
(\!|t+s|\!)^e_\mu & = & (\!|t|\!)^e_\mu + (\!|s|\!)^e_\mu \\
(\!|t-s|\!)^e_\mu & = & (\!|t|\!)^e_\mu - (\!|s|\!)^e_\mu \\
(\!|i*t|\!)^e_\mu & = & \underline{i}\cdot(\!|t|\!)^e_\mu \\
(\!|\lfloor t \rfloor|\!)^e_\mu & = & \underline{\lfloor (\!|t|\!)^e_\mu \rfloor}
\end{array}
\qquad
\begin{array}{lcl}
(\!|\boldsymbol{T}|\!)^e & = & \mathit{True} \\
(\!|\boldsymbol{F}|\!)^e & = & \mathit{False} \\
(\!|s<t|\!)^e & = & (\!|s|\!)^e_\mu < (\!|t|\!)^e_\mu \\
(\!|s>t|\!)^e & = & (\!|s|\!)^e_\mu > (\!|t|\!)^e_\mu \\
(\!|s\leq t|\!)^e & = & (\!|s|\!)^e_\mu \leq (\!|t|\!)^e_\mu \\
(\!|s\geq t|\!)^e & = & (\!|s|\!)^e_\mu \geq (\!|t|\!)^e_\mu \\
(\!|s=t|\!)^e & = & ((\!|s|\!)^e_\mu = (\!|t|\!)^e_\mu) \\
(\!|s\neq t|\!)^e & = & (\!|s|\!)^e_\mu \neq (\!|t|\!)^e_\mu
\end{array}
\qquad
\begin{array}{lcl}
(\!|i \mid t|\!)^e & = & \underline{i} \mid (\!|t|\!)^e_\mu \\
(\!|i \nmid t|\!)^e & = & \underline{i} \nmid (\!|t|\!)^e_\mu \\
(\!|\neg p|\!)^e & = & \neg(\!|p|\!)^e \\
(\!|p\wedge q|\!)^e & = & (\!|p|\!)^e \wedge (\!|q|\!)^e \\
(\!|p\vee q|\!)^e & = & (\!|p|\!)^e \vee (\!|q|\!)^e \\
(\!|p\rightarrow q|\!)^e & = & (\!|p|\!)^e \rightarrow (\!|q|\!)^e \\
(\!|p=q|\!)^e & = & (\!|p|\!)^e \leftrightarrow (\!|q|\!)^e \\
(\!|\exists p|\!)^e & = & \exists x.(\!|p|\!)^{x\cdot e} \\
(\!|\forall p|\!)^e & = & \forall x.(\!|p|\!)^{x\cdot e}
\end{array}
$$

Figure 4.8: Semantics of $\mathcal{R}_{\lfloor\cdot\rfloor}$-terms and formulae

emphasises that qe. is a stronger property. The syntax of terms and formulae is as follows:

$$
\begin{array}{ll}
\mathsf{datatype}\ \mu & = \widehat{\mathbb{Z}} \mid \boldsymbol{x}_{\mathbb{N}} \mid -\mu \mid \mu+\mu \mid \mu-\mu \mid \mathbb{Z}*\mu \mid \lfloor\mu\rfloor \\
\mathsf{datatype}\ \phi & = \mu<\mu \mid \mu>\mu \mid \mu\leq\mu \mid \mu\geq\mu \mid \mu=\mu \mid \mu\neq\mu \mid\ \mathbb{Z} \mid \mu \mid \\
& \quad \mathbb{Z} \nmid \mu \mid \boldsymbol{T} \mid \boldsymbol{F} \mid \neg\,\phi \mid \phi\wedge\phi \mid \phi\vee\phi \mid \phi\rightarrow\phi \mid \phi\leftrightarrow\phi \mid \exists\,\phi \mid \forall\,\phi
\end{array}
$$

The real integer constant $\underline{i}$ in the logic is represented by the term $\widehat{i}$. Bound variables are again represented by de Bruijn indices. We use $\bowtie$ as a place-holder for $=, \neq, <, \leq, >$ or $\geq$ and $\lceil t \rceil$ to denote $-\lfloor -t \rfloor$. The interpretation functions $(\!|.|\!)_\mu$ and $(\!|.|\!)^\cdot$ in Figure 4.8 map the representations back into logic and are parametrised by an environment e (a list of real expressions). Note that in Figure 4.8 we define $(\!|i \mid t|\!)^e$ in terms of the divisibility predicate over the *reals* defined by $x \mid y \leftrightarrow \exists i :: \mathbb{Z}.y = \underline{i}\cdot x$ for $x :: \mathbb{R}$ and $y :: \mathbb{R}$.

Linearity

The fact that $\boldsymbol{x}_0$ does not occur in a μ-term t (resp. in a ϕ-formula p) is formalised by $\mathsf{unbound}_\mu\ t$ (resp. $\mathsf{unbound}\ p$). Substituting t for $\boldsymbol{x}_0$ in p is defined by $p[t]$. Decreasing all variable indexes in p is defined by $\mathsf{decr}\ p$. These functions have such simple recursive definitions that the properties (4.36) are proved automatically.

$$
\begin{array}{r}
\mathsf{unbound}\ p \rightarrow \forall x, y.(\!|p|\!)^{x\cdot e} \leftrightarrow (\!|p|\!)^{y\cdot e} \\
\mathsf{qfree}\ p \rightarrow (\!|p[t]|\!)^{x\cdot e} \leftrightarrow (\!|p|\!)^{((\!|t|\!)^{x\cdot e}_\mu)\cdot e} \\
\mathsf{unbound}\ p \rightarrow \forall x.(\!|\mathsf{decr}\ p|\!)^e \leftrightarrow (\!|p|\!)^{x\cdot e}
\end{array}
\tag{4.36}
$$

We define p to be $\mathcal{R}_+$-linear ($\mathsf{islin}_{\mathcal{R}_+}\ p$) if it is built up from $\wedge, \vee$ and atoms θ either of the form $c * \boldsymbol{x}_0 \bowtie t$, such that $\mathsf{unbound}_\mu\ t \wedge c > 0$, or satisfying $\mathsf{unbound}\ \theta$. We define p to be $\mathcal{Z}_+$-linear in a context e ($\mathsf{islin}_{\mathcal{Z}_+}\ p\ e$) if in addition to the previous requirements every t represents an integer, i.e. $(\!|\lfloor t \rfloor|\!)^e_\mu = (\!|t|\!)^e_\mu$. Moreover $i \mid c * \boldsymbol{x}_0 + t$ and $i \nmid c * \boldsymbol{x}_0 + t$ such that $i > 0 \wedge c > 0 \wedge \mathsf{unbound}_\mu\ t \wedge (\!|t|\!)^e_\mu = (\!|\lfloor t \rfloor|\!)^e_\mu$, are $\mathcal{Z}_+$-linear atoms. A $\mathcal{R}_+$- (resp. $\mathcal{Z}_+$-) linear formula can be regarded as a formula in $\mathcal{R}_+$ (resp. $\mathcal{Z}_+$), assuming $\boldsymbol{x}_0$ will be interpreted by some $x :: \mathbb{R}$ (resp. by some $\underline{i}, i :: \mathbb{Z}$).

Weispfenning's reduction

The main idea is: *"$\lfloor\cdot\rfloor$ is burdensome, get rid of it"*. Since $\forall x.\underline{0} \leq x - \lfloor x \rfloor < \underline{1}$ we have $\exists x.(\!|p|\!)^{x\cdot e} \leftrightarrow \exists i, u.\underline{0} \leq u < \underline{1} \wedge (\!|p|\!)^{(\underline{i}+u)\cdot e}$. Let $\mathsf{split}_0\ p = \widehat{0} \leq \boldsymbol{x}_0 < \widehat{1} \wedge p'$, where $\widehat{0} \leq \boldsymbol{x}_0 < \widehat{1}$ is short for $1 * \boldsymbol{x}_0 + \widehat{0} \geq \widehat{0} \wedge 1 * \boldsymbol{x}_0 + \widehat{-1} < \widehat{0}$, p' results from p by replacing every occurrence of $\boldsymbol{x}_0$ by $\boldsymbol{x}_0 + \boldsymbol{x}_1$ and $\boldsymbol{x}_i$ by $\boldsymbol{x}_{i+1}$ for $i > 0$. We prove

$$
\mathsf{qfree}\ p \rightarrow ((\!|\exists p|\!)^e \leftrightarrow \exists i, u.(\!|\mathsf{split}_0\ p|\!)^{\underline{i}\cdot u\cdot e}) \tag{4.37}
$$

One main contribution of [Wei99] is to supply two functions $\mathsf{lin}_{\mathcal{R}}$ and $\mathsf{lin}_{\mathcal{Z}}$, which transform any qf. p into a $\mathcal{R}_+$- (resp. $\mathcal{Z}_+$-) linear formula (cf. (4.39) and (4.38)), assuming that $\boldsymbol{x}_0$ is interpreted by $u \in [\underline{0}, \underline{1})$ (resp. by $\underline{i}$).

$$\mathsf{qfree}\ p \to (\!|\mathsf{lin}_{\mathcal{Z}}\ p|\!)^{\underline{i}\cdot e} \leftrightarrow (\!|p|\!)^{\underline{i}\cdot e} \wedge \mathsf{islin}_{\mathcal{Z}_+}\ (\mathsf{lin}_{\mathcal{Z}}\ p)\ (\underline{i}\cdot e) \tag{4.38}$$

$$\mathsf{qfree}\ p \wedge \underline{0} \le x < \underline{1} \to (\!|\mathsf{lin}_{\mathcal{R}}\ p|\!)^{x\cdot e} \leftrightarrow (\!|p|\!)^{x\cdot e} \wedge \mathsf{islin}_{\mathcal{R}_+}\ (\mathsf{lin}_{\mathcal{R}}\ p) \tag{4.39}$$

We subsequently exhibit $\mathsf{lin}_{\mathcal{Z}}$ and $\mathsf{lin}_{\mathcal{R}}$, which mainly *"get rid of* $\lfloor\cdot\rfloor$*"*. Now given two qe. procedures $qe_{\mathcal{R}_+}$ for $\mathcal{R}_+$ and $qe_{\mathcal{Z}_+}$ for $\mathcal{Z}_+$ satisfying the equations below, it is simple to prove that $\mathsf{mir} = qe_{\mathcal{Z}_l} \circ \mathsf{lin}_{\mathcal{Z}} \circ qe_{\mathcal{R}_l} \circ \mathsf{lin}_{\mathcal{R}} \circ \mathsf{split}_0$ eliminates $\exists$ over a qf. body.

$$\mathsf{islin}_{\mathcal{R}_+}\ p \to (\!|qe_{\mathcal{R}_+}\ \widehat{0} \le \boldsymbol{x}_0 < \widehat{1} \wedge p|\!)^e \leftrightarrow (\!|\exists\, \widehat{0} \le \boldsymbol{x}_0 < \widehat{1} \wedge p|\!)^e \wedge \mathsf{qfree}(qe_{\mathcal{R}_+}\ p)$$
$$\mathsf{islin}_{\mathcal{Z}_+}\ p \to (\!|qe_{\mathcal{Z}_+}\ p|\!)^e \leftrightarrow \exists i.(\!|p|\!)^{\underline{i}\cdot e} \wedge \mathsf{qfree}(qe_{\mathcal{Z}_+}\ p)$$

$\mathsf{lin}_{\mathcal{Z}}$

In order to define $\mathsf{lin}_{\mathcal{Z}}$ and prove (4.38), we first introduce a function $\mathsf{split}_{\mathcal{Z}}$ that, given a $t :: \mu$, returns an integer c and $s :: \mu$ (not involving $\boldsymbol{x}_0$), such that (4.40) (Lemma 3.2 in [Wei99]) holds. Note that $\boldsymbol{x}_0$ is interpreted by a real integer $\underline{i}$.

$$\mathsf{split}_{\mathcal{Z}}\ t = (c, s) \to (\!|c * \boldsymbol{x}_0 + s|\!)^{\underline{i}\cdot e}_{\mu} = (\!|t|\!)^{\underline{i}\cdot e}_{\mu} \wedge \mathsf{unbound}_{\mu} s \tag{4.40}$$

The definition of $\mathsf{split}_{\mathcal{Z}}$ and the proof of (4.40) proceed by induction on t. If $t = \lfloor t' \rfloor$ then return $(c, \lfloor s \rfloor)$, where $\mathsf{split}_{\mathcal{Z}}\ t' = (c, s)$. Remember that $\lfloor x + \underline{j} \rfloor = \lfloor x \rfloor + j$ holds for any $j :: \mathbb{Z}$. The other cases are trivial.

Now $\mathsf{lin}_{\mathcal{Z}}$ is simple: push negations inwards and transform atoms according to the result of $\mathsf{split}_{\mathcal{Z}}$ and the properties (4.41), cf. example 4.3.3 for the $=$ case, where the first property in (4.41) is used. By induction on p, we easily prove (4.38) using the properties (4.40) and (4.41).

$$\begin{aligned} \underline{c\cdot i} = y &\leftrightarrow (c\cdot i = \lfloor y \rfloor \wedge y = \underline{\lfloor y \rfloor}) \\ \underline{c\cdot i} < y &\leftrightarrow (c\cdot i < \lfloor y \rfloor \vee c\cdot i = \lfloor y \rfloor \wedge \underline{\lfloor y \rfloor} < y) \\ \underline{d} \mid \underline{c\cdot i} + y &\leftrightarrow (\underline{\lfloor y \rfloor} = y \wedge d \mid c\cdot i + \lfloor y \rfloor) \\ \underline{0} \mid x &\leftrightarrow (x = \underline{0}) \end{aligned} \tag{4.41}$$

Example

$$\begin{aligned} \mathsf{lin}_{\mathcal{Z}}\ (t = t') = \quad & \textbf{let}\ (c, s) = \mathsf{split}_{\mathcal{Z}}\ (t - t')\ \textbf{in} \\ & \textbf{if}\ c = 0\ \textbf{then}\ s = \widehat{0} \\ & \textbf{else if}\ c > 0\ \textbf{then}\ c * \boldsymbol{x}_0 + \lceil s \rceil = \widehat{0} \wedge \lfloor s \rfloor - s = \widehat{0} \\ & \textbf{else}\ -c * \boldsymbol{x}_0 + \lfloor -s \rfloor = \widehat{0} \wedge \lfloor s \rfloor - s = \widehat{0} \end{aligned}$$

$\mathsf{lin}_{\mathcal{R}}$

In order to define $\mathsf{lin}_{\mathcal{R}}$ and prove (4.39), we first introduce a function $\mathsf{split}_{\mathcal{R}} : \mu \Rightarrow [\boldsymbol{\phi} \times \mathbb{Z} \times \mu]$ which, given a $t :: \mu$, yields a *complete* finite case distinction given by $\mathcal{R}_+$-linear formulae ϕ_i and corresponding μ-terms s_i (not involving $\boldsymbol{x}_0$) and integers c_i such that $(\!|t|\!)^{u\cdot e}_{\mu} = (\!|c_i * \boldsymbol{x}_0 + s_i|\!)^{u\cdot e}_{\mu}$ whenever $(\!|\phi_i|\!)^{u\cdot e}$ holds (Lemma 3.3 in [Wei99]), i.e.

$$\begin{aligned} \forall(\phi_i, c_i, s_i) \in \{\!\{\mathsf{split}_{\mathcal{R}}\ t\}\!\}.((\!|\phi_i|\!)^{u\cdot e} \to (\!|t|\!)^{u\cdot e}_{\mu} = (\!|c_i * \boldsymbol{x}_0 + s_i|\!)^{u\cdot e}_{\mu}) & \\ \wedge \mathsf{unbound}_{\mu}\ s_i \wedge \mathsf{islin}_{\mathcal{R}_+}\ \phi_i & \quad (4.42) \\ \underline{0} \le u < \underline{1} \to \exists(\phi_i, c_i, s_i) \in \{\!\{\mathsf{split}_{\mathcal{R}}\ t\}\!\}.(\!|\phi_i|\!)^{u\cdot e} & \quad (4.43) \end{aligned}$$

These cases are also disjoint as noted in [Wei99], but this is only needed for complexity considerations. The definition of $\mathsf{split}_{\mathcal{R}}$ and the proof of (4.42) and (4.43) proceed by induction on t. Assume $t = \lfloor t' \rfloor$, let $(\phi'_i, c'_i, s'_i) \in \{\!\{\mathsf{split}_{\mathcal{R}}\ t'\}\!\}$ and assume $(\!|\phi'_i|\!)^{u\cdot e}$ and $c'_i > 0$. By the

induction hypothesis $(\!|t'|\!)_\mu^{u\cdot e} = (\!|c'_i * \boldsymbol{x}_0 + s'_i|\!)_\mu^{u\cdot e}$ holds and since $\underline{0} \le u < \underline{1}$ and $c'_i > 0$, it follows that $j \le c'_i \cdot u < j+1$ for some $j \in \{0 \ldots c'_i\}$, i.e.

$$\underline{j + \lfloor (\!|s'_i|\!)_\mu^{u\cdot e} \rfloor} \le (\!|c'_i * \boldsymbol{x}_0 + s'_i|\!)_\mu^{u\cdot e} < \underline{j+1 + \lfloor (\!|s'_i|\!)_\mu^{u\cdot e} \rfloor}$$

and hence $\lfloor (\!|c'_i * \boldsymbol{x}_0 + s'_i|\!)_\mu^{u\cdot e} \rfloor = j$. For $(\phi'_i, c'_i, s'_i) \in \{\!\!\{\mathsf{split}_\mathcal{R}\ t'\}\!\!\}$, $\mathsf{split}_\mathcal{R}$ returns the list of $(\phi'_i \wedge A_j, 0, \lfloor s \rfloor + \widehat{j})$, where $j \in \{0 \ldots c'_i\}$, $A_j = r \ge \widehat{j} \wedge r < \widehat{j+1}$ and $r = c'_i * \boldsymbol{x}_0 + s'_i - \lfloor s'_i \rfloor$. The cases $c'_i < 0$ and $c'_i = 0$, ignored in [Wei99], are analogous. The other cases for t are simple.

The definition of $\mathsf{lin}_\mathcal{R}$ is involved for atoms, but very simple for the rest: it just pushes negations inwards. Due to the result of $\mathsf{split}_\mathcal{R}$, assume that atoms have the form $f(c * \boldsymbol{x}_0 + s)$, where s does not involve $\boldsymbol{x}_0$ and $f \in \{\lambda t.t \bowtie \widehat{0}, \lambda t.i \mid t, \lambda t.i \nmid t \text{ for some } i\}$. For every f, we define its corresponding $\mathcal{R}_+$-*linear* version $f_l : \mathbb{Z} \Rightarrow \mu \Rightarrow \boldsymbol{\phi}$, and prove (4.44). Example 4.3.3 shows the case for $=$ and the corresponding definition of $\mathsf{lin}_\mathcal{R}$.

$$\begin{aligned} \underline{0} \le u < \underline{1} \wedge \mathsf{unbound}_\mu\ s \wedge (\!|t|\!)_\mu^{u\cdot e} = (\!|c * \boldsymbol{x}_0 + s|\!)_\mu^{u\cdot e} \\ \rightarrow ((\!|f_l\ c\ s|\!)^{u\cdot e} \leftrightarrow (\!|f\ t|\!)^{u\cdot e}) \wedge \mathsf{islin}_{\mathcal{R}_+}\ (f_l\ c\ s) \end{aligned} \tag{4.44}$$

Example

$$\begin{aligned} c * \boldsymbol{x}_0 + s =_{\boldsymbol{l}} \widehat{\boldsymbol{0}} = \quad & \textbf{if } c = 0 \textbf{ then } s = \widehat{0} \textbf{ else} \\ & \textbf{if } c > 0 \textbf{ then } c * \boldsymbol{x}_0 + s = \widehat{0} \textbf{ else } -c * \boldsymbol{x}_0 + - s = \widehat{0} \end{aligned}$$

$$\begin{aligned} \mathsf{lin}_\mathcal{R}(t = t') = \quad & \textbf{let } [(p_0, c_0, s_0), ..., (p_n, c_n, s_n)] = \mathsf{split}_\mathcal{R}\ (t - t') \\ & \textbf{in } (p_0 \wedge (c_0 * \boldsymbol{x}_0 + s_0 =_{\boldsymbol{l}} \widehat{\boldsymbol{0}})) \vee \ldots \vee (p_n \wedge (c_n * \boldsymbol{x}_0 + s_n =_{\boldsymbol{l}} \boldsymbol{0})) \end{aligned}$$

Since $\mid$ and $\nmid$ are not $\mathcal{R}_+$-linear, their corresponding linear versions eliminate them at the cost of a case distinction according to (4.45).

$$\begin{aligned} \underline{0} \le u < \underline{1} \wedge c > 0 \rightarrow \\ (\underline{d} \mid \underline{c} \cdot u + s \leftrightarrow \exists j \in \{0..c-1\}.(\underline{c} \cdot u = \underline{j} + \underline{\lceil s \rceil} - s) \wedge d \mid j + \lceil s \rceil) \end{aligned} \tag{4.45}$$

Note here the importance of the coefficients being *integers*. Of course rational coefficients add no expressibility power to the theory and can be eliminated to integer coefficients but admitting multiplication by arbitrary real values yields undecidability, see [Wei99] for a very beautiful proof by encoding irrationality of a number in the new language. We implement the case distinction in (4.45) by dvd, and $\mid_l$ follows naturally:

$$\begin{aligned} d\ \mathsf{dvd}\ c * \boldsymbol{x}_0 + s = \quad & (c * \boldsymbol{x}_0 + s - \lceil s \rceil - \widehat{0} = \widehat{0} \wedge d \mid \lceil s \rceil + \widehat{c-1}) \vee \ldots \\ & \vee (c * \boldsymbol{x}_0 + s - \lceil s \rceil - \widehat{c-1} = \widehat{0} \wedge d \mid \lceil s \rceil + \widehat{c-1}) \\ d \mid_l c * \boldsymbol{x}_0 + s = \quad & \textbf{if } d = 0 \textbf{ then } c * \boldsymbol{x}_0 + s =_{\boldsymbol{l}} \boldsymbol{0} \textbf{ else} \\ & \textbf{if } c = 0 \textbf{ then } d \mid s \textbf{ else} \\ & \textbf{if } c > 0 \textbf{ then } |d|\ \mathsf{dvd}\ c * \boldsymbol{x}_0 + s \\ & \textbf{else } |d|\ \mathsf{dvd}\ -c * \boldsymbol{x}_0 + - s \end{aligned}$$

Now we define $\mathsf{lin}_\mathcal{R}(d \mid t)$ analogously to the $=$-case.

$$\begin{aligned} \mathsf{lin}_\mathcal{R}(d \mid t) = \quad & \textbf{let } [(p_0, c_0, s_0), ..., (p_n, c_n, s_n)] = \mathsf{split}_\mathcal{R}\ t\ ; \\ & \quad g = \lambda c, s.d \mid_l c * \boldsymbol{x}_0 + s \\ & \textbf{in } (p_0 \wedge (g\ c_0\ s_0)) \vee \ldots \vee (p_n \wedge (g\ c_n\ s_n)) \end{aligned}$$

Note that $\mathsf{lin}_\mathcal{R}$ has akin definitions for the atoms. In fact for an atom $f(t)$, the actual definition is $\mathsf{lin}_\mathcal{R}(f(t)) = \mathsf{split}_l\ f_l\ t$, where $\mathcal{R}_+$-linear version of f.

$$\begin{aligned} \mathsf{split}_l\ f_l\ t = \quad & \textbf{let } [(p_0, c_0, s_0), \ldots, (p_n, c_n, s_n)] = \mathsf{split}_\mathcal{R}\ t \\ & \textbf{in } (p_0 \wedge (f_l\ c_0\ s_0)) \vee \ldots \vee (p_n \wedge (f_l\ c_n\ s_n)) \end{aligned}$$

We prove the following simple, yet generic property for split_l

$$\begin{aligned} \underline{0} \leq u < \underline{1} \wedge (\forall t, c, s.\mathsf{unbound}_\mu\ s \wedge (\!|t|\!)_\mu^{u\cdot e} = (\!|c * \boldsymbol{x}_0 + s|\!)_\mu^{u\cdot e} \\ \rightarrow (\!|f_l\ c\ s|\!)^{u\cdot e} \leftrightarrow (\!|f\ t|\!)^{u\cdot e} \wedge \mathsf{islin}_{\mathcal{R}_+}\ (f_l\ c\ s)) \\ \rightarrow \mathsf{islin}_{\mathcal{R}_+}\ (\mathsf{split}_l\ f_l\ t) \wedge ((\!|\mathsf{split}_l\ f_l\ t|\!)^{u\cdot e} \leftrightarrow (\!|f\ t|\!)^{u\cdot e}) \end{aligned} \tag{4.46}$$

Note that the premise of (4.46), which expresses that f_l is a $\mathcal{R}_+$-linear version of f, is discharged by the instances of (4.44) for each different f. After all these preparations, it is not surprising that we proved (4.39) automatically. The formalisation of the full qep. for $\mathcal{R}_{\lfloor\cdot\rfloor}$ needed 4000 lines of Isabelle proofs, but including a qep. for Presburger arithmetic and linear real arithmetic.

Definable sets

We already know that the definable sets in $\mathcal{Z}_+$ are the almost periodic sets and those of $\mathcal{R}_+$ are finite unions of intervals with rational or infinite endpoints. Call these last sets *simple*. The definable sets in $\mathcal{R}_{\lfloor\cdot\rfloor}$ have a mixed structure of the above sets. Let $A \subseteq \mathbb{R}$ be *periodically simple* if $A = \bigcup_{n=-\infty}^{+\infty} np + B$, for some rational number p and a simple set $B \subseteq [0, p)$. Let $A \subseteq \mathbb{R}$ be *ultimately periodically simple* if $A = (A' \setminus B) \cup C$ for a periodically simple set A' and simple sets B and C. $A \subseteq \mathbb{R}$ is definable in $\mathcal{R}_{\lfloor\cdot\rfloor}$ if and only if it is ultimately periodically simple. See [Wei99] for a very beautiful proof.

4.4 Algebraically closed fields

> Der Heilige Geist fand einen erhabenen Ausweg in der Analysis mit diesem Mittelding zwischen Sein und Nichtsein, das wir als imaginäre (Quadrat)wurzel der negativen Einheit bezeichnen.
>
> *(G. W. Leibniz)*

The theory of algebraically closed fields (ACF) extends the theory of fields by the axiom for algebraic closure: every non-constant polynomial has a root. In the following we present a proof-producing qep. *only* for the complex numbers $\mathbb{C}$, based on [KK67]. First, we give a formalisation of the fundamental theorem of algebra in §4.4.1. In §4.4.2, we present the qep. abstractly. Finally we sketch a derived rule in §4.4.3.

4.4.1 The fundamental theorem of algebra

The fundamental theorem of algebra states that $\mathbb{C}$ is algebraically closed. Let $f :: \alpha \Rightarrow \beta$ be constant ($\mathsf{constant}\ f$) if $\forall x, y.f\ x = f\ y$. Then we formalise the fundamental theorem of algebra in (4.47).

$$\neg\mathsf{constant}\ \overline{p} \rightarrow \exists z :: \mathbb{C}.\overline{p}\ z = 0 \tag{4.47}$$

Proof of (4.47). We present Argrand's proof [Ebb04, §4.2] as refined in [Lit41, Est56] and the ingenious argument in [Har01], which avoids using $\sqrt[n]{\cdot}$ for complex numbers and $n \neq 2$. The main property to start with, and which we prove later, is that the complex modulus of any non-constant complex polynomial attains an overall minimum:

$$\forall p.\exists z.\forall w.|\overline{p}\ z| \leq |\overline{p}\ w| \tag{4.48}$$

The proof of (4.47) is now by complete induction on the degree n of p. By (4.48) we know that $\forall w.|\overline{p}\ c| \leq |\overline{p}\ w|$ holds for some c. Assume for contradiction that $\overline{p}\ c \neq 0$ and consider the polynomials q and r such that $|p| = |q| = |r| = n$ and $\forall z.\overline{q}\ z = \overline{p}\ (z + c)$ and $\forall z.\overline{r}\ z = \frac{1}{\overline{q}\ 0} \cdot \overline{q}\ z$. Note that $\overline{q}$ and $\overline{r}$ are not constant and that $\overline{q}\ 0 = \overline{p}\ c \neq 0$ and that $\overline{r}\ 0 = 1$. To derive a contradiction it is sufficient to prove $|\overline{r}\ w| < 1$ for some w, since $\forall w.|\overline{r}\ w| < 1 \leftrightarrow |\overline{q}\ w| < |\overline{q}\ 0|$. For this we decompose r such that $\forall z.\overline{r}\ z = \overline{r}\ 0 + z^k \cdot \overline{a \cdot s}\ z$,

for $a \neq 0, k \neq 0$ and $|s| + k + 1 = |r| = n$. That such a decomposition is possible for any non-constant polynomial p is a simple proof by induction on p. Note that $k + 1 \leq n$. First consider the case where $k + 1 < n$, then by the induction hypothesis, property (4.47) holds for the polynomial $1 + a \cdot z^k$, i.e. $1 + a \cdot w^k = 0$ for some w. After several technical manipulations, we show that $|\overline{r}\ w| < 1$. In the case $k + 1 = n$ we have $s = []$. Hence $\forall z.|r(z)| = |1 + a \cdot z^k|$ and the claim is an immediate consequence of (4.49).

$$\forall b, n.b \neq 0 \wedge n \neq 0 \rightarrow \exists z.|1 + b \cdot z^n| < 1 \tag{4.49}$$

□

Proof of (4.49). The proof of (4.49) is by complete induction over n. If n is even, then we simply apply the induction hypothesis to $\frac{n}{2}$ and use the complex square root function. For the case where n is odd, first note that $\forall z.|z| = 1 \rightarrow |z+1| < 1 \vee |z-1| < 1 \vee |z-i| < 1 \vee |z+i| < 1$. We apply this property to $z = \frac{|b|}{b}$ and hence obtain $|\frac{|b|}{b} + v^k| < 1$ for some v (note that $w \in I \rightarrow w^k \in I$ for $I = \{1, -1, i, -i\}$). Now (4.49) holds for $\frac{v}{\sqrt[k]{|b|}}$. □

Proof of (4.48). This proof is simple once the key properties (4.50) and (4.51) are proved. The first states that the modulus of a complex polynomial goes to infinity, when the modulus of its argument does. The second (4.51) states that the complex module of polynomials attains a minimum over discs. This is a consequence of the continuity of polynomials (cf. (4.52)) and the topological compactness of complex discs, cf. (4.53) for the Bolzano-Weierstrass formulation.

$$(\exists c \in \{\!\{p\}\!\}.c \neq 0) \rightarrow \forall a, d.\exists r.\forall z.|z| \geq r \rightarrow |\overline{a \cdot p}\ z| \geq d \tag{4.50}$$

$$\forall p, r.\exists z.\forall w.|w| \leq r \rightarrow |\overline{p}\ z| \leq |\overline{p}\ w| \tag{4.51}$$

$$\forall p, z.\forall e > 0.\exists d > 0.\forall w.w < |w - z| < d \rightarrow |\overline{p}\ w - \overline{p}\ w| < e \tag{4.52}$$

$$(\forall n :: \mathbb{N}.|s\ n| \leq r) \rightarrow \exists f z.(\forall m, n.m < n \rightarrow f\ m < f\ n) \wedge$$
$$\forall e > 0.\exists N.\forall n \geq N.|s(f\ n) - z| < e \tag{4.53}$$

□

Proof of (4.52). Fix p, z and $e > 0$ and consider a polynomial q such that $|q| = |p|$ and $\forall x.\overline{q}\ x = \overline{q}\ (z + x)$. The proof of (4.52) is by induction on q. The [] case is trivial. For $q = a \cdot h$ we know that $\forall z.|z| \leq 1 \rightarrow |\overline{h}\ z| \leq m$ for some $m > 0$ and that there is a d such that $0 < d < 1 \wedge d < \frac{e}{m}$. Now (4.52) holds for d. □

Proof of (4.53). Let s and r be as in the premise or (4.53) and call f a sub-sequence if $\forall m < n.f\ m < f\ n$. Moreover let $\Re$ and $\Im$ denote the real and imaginary part functions. Then we know that there are two sub-sequences f and g such that $\Re \circ s \circ f$ and $\Im \circ s \circ f \circ g$ are monotonic and hence converge to some x and y respectively. Now (4.53) holds for $h = f \circ g$ and $z = x + i \cdot y$, where $i^2 = -1$. □

4.4.2 A quantifier elimination procedure

We only present the qe. for the following case (cf. §4.1 for this to be sufficient)

$$\exists z. \bigwedge_{i=1}^{n} p_i(z) = 0 \wedge \bigwedge_{j=1}^{m} q_j(z) \neq 0. \qquad (*)$$

First note that $(*)$ is equivalent to $\exists z. \bigwedge_{i=1}^{n} p_i(z) = 0 \wedge q(z) \neq 0$, where $q = \prod_{j=1}^{m} q_i$. If $n = 0$ then q is either a constant non-zero polynomial or it takes infinitely many values and hence $q(z) \neq 0$ for some z. If $n \geq 1$ then pick some equation $p = 0$ where p has minimal degree d, i.e. $p(z) = a \cdot z^d + r(z)$, for some r with degree $l < d$. If $a = 0$, then proceed recursively replacing $p = 0$ by $r = 0$. Otherwise $a \neq 0$. If $n > 1$ then reduce the degree of

the other equations by pseudo-division by p. For a given p_i we have $a^k \cdot p_i = p \cdot s + t$ for some k and polynomials s and t. Since $a \neq 0$ and $p = 0$ we have $p_i = 0 \leftrightarrow t = 0$. If $n = 1 \wedge m = 0$ then, by the algebraic closure, an equivalent qf. formula is that at least one coefficient of p must be non-zero. Finally if $n = 1 \wedge m \neq 0$, then we have $\exists z.p(z) = 0 \wedge q \neq 0$ is equivalent to $\neg\forall z.p(z) = 0 \rightarrow q(z) = 0$. Moreover, $\forall z.p(z) = 0 \rightarrow q(z) = 0$ (i.e. all zeros of p are zeros of q) is equivalent to $p|q^d \vee p = q = 0$, where d is the degree of p. Note that taking q^d is necessary, since a zero of p might have multiplicity greater than one (yet never greater than d) in p but one in q. The second disjunct is formally necessary, since if $p = 0$ then $d = 0$ and hence $q^d = 1$ and $0|1$ is not equivalent to $\forall z.q(z) = 0$, if $q \neq 0$. Hence we only need to encode $p|q^d$ by a qf. formula. For this pseudo-divide q^d by p and obtain $a^k \cdot q^d = p \cdot s + t$, for some k and polynomials s and t. Since $a \neq 0$ we have $p|q^d \leftrightarrow p|t$. The latter holds only if t is the zero polynomial, since it has a degree less than d. Finally $p|q^d$ is equivalent to $\bigwedge_{i=1}^{l} c_i = 0$, where the c_i's are the coefficients of t.

Definable sets

Atomic formulae $p = 0$ and $q \neq 0$ define a finite set of algebraic numbers and the complement of such a set, respectively. Hence by qe. any formula in ACF defines a finite set or the complement of a finite set of algebraic numbers. Conversely such sets are also definable in ACF and ACF is strongly minimal.

More precisely let K be an algebraically closed field and consider $S \subseteq K[x_1, \ldots, x_n]$ (S can be infinite). Denote the affine variety of S by $V(S) = \{\vec{x} \in K^n | p(\vec{x}) = 0 \text{ for all } p \in S\}$ and call $A \subseteq K^n$ Zariski closed if $A = V(S)$ for some S. Zariski closed sets are closed under finite unions and arbitrary intersections (using Hilbert's basis theorem [Hil90]) and are hence the closed sets of a topology (Zariski topology). $A \subseteq K^n$ is definable in ACF if and only if it is a boolean combination of Zariski closed sets.

4.4.3 A derived rule

We adopted a derived rule by Harrison [Har01] in HOL Light. We do not present any details here, but emphasise the key theorems needed and then give an overview of the procedure.

The key theorems The algorithm in §4.4.2 is mainly based on two theorems: the fundamental theorem of algebra and the reduction of the case of one equation and one inequation to a divisibility of two polynomials. We have already proved the fundamental theorem of algebra in (4.47). For the derived rule we need a syntactical criterion for non-constant polynomials. The variant in (4.54) is easily derivable from (4.47). The second important theorem (4.55) has a Nullstellensatz flavour. The proof of (4.55) is by induction on $\mathsf{deg}\ p$.

$$(\forall c, cs.p = c \cdot cs \rightarrow (c = 0 \vee \exists b \in \{\!\{cs\}\!\}.b \neq 0)) \rightarrow \exists x :: \mathbb{C}.\overline{p}\ z = 0 \tag{4.54}$$

$$(\forall z.\overline{p}\ z = 0 \rightarrow \overline{q}\ z = 0) \leftrightarrow (p \mid q^{\mathsf{deg}\ p} \vee \overline{p} = \overline{[]} \wedge \overline{q} = \overline{[]}) \tag{4.55}$$

The general structure The main loop of the derived rule proceeds exactly as explained in §4.4.2. To organise case splits over the heads of polynomials we use continuations. We can think of a continuation as a specific instance of the whole qep., using some assumptions. After several case-splits, we arrive at one of the basic cases as presented in §4.4.2. We only sketch the case of one equation and one inequation, i.e. $\exists z.\overline{p}\ z = 0 \wedge \overline{q}\ z \neq 0$. Using (4.55) we reduce the problem to dealing with the divisibility statement $p \mid q^{\mathsf{deg}\ p}$. Note that by (3.7), we reduce $\overline{p} = \overline{[]}$ and $\overline{q} = \overline{[]}$ to conditions on their coefficients. To obtain a qf. equivalent to $p \mid q^{\mathsf{deg}\ p}$, we perform pseudo-division as in §4.4.2, and find a pseudo-remainder r. We use (4.56) to obtain an equivalent formulation in terms of the pseudo-remainder (in (4.56), a is the head of p and p' the pseudo quotient). The qf. equivalent is an immediate application

of (4.57). Of course, when we compute the pseudo-remainder correctly, then $\mathsf{deg}\ r < \mathsf{deg}\ p$ holds.

$$a \neq 0 \wedge p \mid p' \wedge (\forall z. a \cdot \overline{q}\ z - \overline{p'}\ z = \overline{r}\ z) \rightarrow (p \mid q \leftrightarrow p \mid r) \tag{4.56}$$

$$p \mid q \rightarrow \mathsf{deg}\ p \leq \mathsf{deg}\ q \vee \overline{q} = \overline{[]} \tag{4.57}$$

For the formalisation of this procedure we needed 1300 lines of Isabelle proofs, including the fundamental theorem of algebra, and further 800 lines of SML code for the qe. conversion.

4.5 Real closed fields

The theory of real closed fields (RCF) is the theory of ordered fields having the intermediate value property, but see [BPR03] for many equivalent formulations. It is not difficult to show that RCF over the language of rings does not admit qe. Consider $P(c) = \exists x. x^2 + 2 \cdot x + c = 0$. Then P defines the set $\mathcal{S}(P) = \{c \mid c \leq 1\}$, which can not be expressed in the qf. language of rings. We must hence add ordering to the language of rings. The qe. result for RCF by Tarski [Tar51] is considered one of the most important results in formal logic of the last century.

4.5.1 A quantifier elimination procedure

Consider $\exists x. F(x)$, where F is qf. over the language of RCF. F consists of logical connectives ($\wedge, \vee, \rightarrow, \leftrightarrow$ and $\neg$) over atoms $p_i \bowtie 0$ for some polynomial p_i and $\bowtie \in \{=, \neq, \leq, \geq, <, >\}$. Let $At(F) = \bigcup_{i \in I} \{p_i \bowtie_i 0\}$ denote all non-trivial atoms of F. Assume that we know the signs of all polynomials p_i over a set $S \neq \emptyset$ for all $i \in I$, i.e. we have a mapping $\rho_S : I \rightarrow \{=, <, >\}$ such that $\forall x \in S. p_i(x)\ \rho_S(i)\ 0$ for $i \in I$. Note that using ρ_S we can "evaluate" F over S (denoted by $F[\rho_s]$) to *True* or *False* and hence decide $\exists x \in S. F(x)$ in particular. We call ρ_S a *sign environment* over S (or short SE over S or just SE). The idea of the qep. is to partition the real line into disjoint non-empty sets $S_1, \ldots, S_k$ (they are in fact intervals) and compute a SE for each S_i. This is a two dimensional mapping and we refer to it by a *sign matrix* (or SM). The main challenge for the qep. is that any of the S_i's is expressible by an RCF formula ψ_i not involving the bound variable x. Taking the liberty to write $F[\psi_i]$ for $F[S_i]$, we ultimately obtain

$$\exists x. F(x) \leftrightarrow \bigvee_{i=1}^{k} \psi_i \wedge F[\psi_i].$$

The implication from right to left is clear, and the other direction relies on the fact that partitioning the real line into intervals S_i is exhaustive.

Computing the sign matrix

The basic case is that we have a constant polynomial c, i.e. not involving the bound variable x. Here we just split over all possible signs of c (of course avoiding trivial assumptions like $4 < 0$). The signs in the SM can be zero, positive or negative. A SM represents a "formula" encoding our assumptions for further steps in the algorithm. In fact we compute a SM depending on another SM, which represents the signs *known so far*. The intervals in the SM describe a *partition* of $\mathbb{R}$ into non-empty intervals and points. We can think of them as $z_1 < \cdots < z_m$. The z_j's turn out to be zeros of the polynomials under consideration. We need no numerical value for $z_1, \ldots, z_m$, but just their existence and their *ordered positions* on the real line. A SM assigns a sign to each polynomial over each point $\{z_j\}$ and interval $]z_j, z_{j+1}[$ for every $0 \leq j \leq m$ and $z_0 = -\infty$ and $z_{m+1} = +\infty$.

The crucial observation in the algorithm is that to determine a SM for $p, p_1, \ldots, p_n$ we just need to find one for $p_0, p_1, \ldots, p_n, r_0, \ldots, r_n$, where $p_0 = p'$ is the first derivative of p with respect to x and r_i is the pseudo-remainder of p by p_i for $0 \leq i \leq n$. Assuming that a

is the head of p, we have $a^k \cdot p(x) = q_i(x) \cdot p_i(x) + r_i(x)$ for some k and polynomial $q_i(x)$, for $0 \leq i \leq n$. Taking p with the largest degree in x among $p, p_1, \ldots, p_n$, it is clear that by proceeding as above we reduce the highest degree in x (or at least if there are several candidates for p, we reduce their number). Recall that $\partial r_i < \partial p$, for $0 \leq i \leq n$. More precisely, given p as above, we consider the two cases $a = 0$ and $a \neq 0$. If $a = 0$, then we recur removing p's head and hence reducing its degree. If $a \neq 0$, then we determine a SM for $p_0, \ldots, p_n, r_0, \ldots, r_n$ as above. Assume we have such a SM over say $z_1 < \cdots < z_m$ and consider a SE in it. First, we can derive the sign of p at the zeros of $p_0, \ldots, p_n$. Recall that $a^k \cdot p(x) = q(x) \cdot p_i(x) + r_i(x)$ for every $0 \leq i \leq n$ and hence at a zero z_l of some p_j the sign of p is exactly the sign of r_j, if a is positive or k is even, and exactly the opposite sign of r_j, if k is odd and a is negative. For all intervals and other points which are not zeros of some p_i we assign "non-zero" sign and will infer the precise signs shortly. This was actually all we needed from the SE for $r_0, \ldots, r_n$ and can throw it away. Moreover we remove all points where at least one r_j is zero but none of the p_j's for $0 \leq j \leq n$. Note that the signs of $p_0, \ldots, p_n$ are not altered by the removed points, since none of the p_i's is zero and hence can change its sign. The signs of $p_0, \ldots, p_n$ over the new merged intervals can hence be read from any of the sub-intervals.

For now we have a SE for $p_0, \ldots, p_n$, notably we know *all* zeros of $p_0, \ldots, p_n$ and the exact sign of p at these zeros. Hence it remains to derive potential zeros of p and its sign over intervals. This is not difficult since we have the exact signs of $p_0 = p'$ and the intermediate value theorem at hand. Consider an interval $]w_i, w_{i+1}[$ and recall that p' has no zeros over $]w_i, w_{i+1}[$, then p has at *most* one zero in $]w_i, w_{i+1}[$. We can decide this by comparing the signs of p at w_i and w_{i+1} respectively: they must not be zero and must be different. If such a zero exists then it is a zero only of p and we introduce a new point u between w_i and w_{i+1} where p has sign zero at u, the same sign as at w_i and w_{i+1} over $]w_i, u[$ and $]u, w_{i+1}[$ respectively. For the other polynomials we duplicate their signs over the new point and intervals. If no such zero exists, then the sign of p over $]w_i, w_{i+1}[$ is one of its sign at w_i or w_{i+1}, which is not zero. Note that at least one is not zero, since otherwise p' would have a zero over $]w_i, w_{i+1}[$.

Definable sets

A subset of $\mathbb{R}$ is definable in RCF if and only if it is a finite union of intervals, whose endpoints are algebraic or infinite. A subset S of $\mathbb{R}^n$ is definable in RCF if and only if it is semi-algebraic, i.e. $S = \bigcup_{i=1}^{k} \bigcap_{j=1}^{m_i} \{\vec{x} \in \mathbb{R}^n \mid p_{ij}(\vec{x}) \bowtie_{ij} 0\}$, where $p_{ij}(\vec{x}) \in \mathbb{R}[\vec{x}]$ and $\bowtie_{ij} \in \{=, >\}$. See [BR90] for an overview of semi-algebraic sets and their properties.

4.5.2 A derived rule

In order to give a rough idea about the implementation of a derived rule for the algorithm in §4.5.1, we first present some more polynomials utilities (derivatives) and an encoding of SE and SM. We subsequently only sketch the main theorems needed for the procedure and its main loop. We present more implementation details in §4.5.3.

More polynomial utilities

For the implementation of the derived rule we use the univariate polynomials instantiated to the reals. For $\exists x.P(x)$, where P is qf., we consider polynomials occurring in P as univariate in x, where the coefficients are polynomials in variables, which are free or bound by outer quantifiers. We order these variables, say x, y, z etc, so that the coefficients of a polynomial in x are polynomials in y, whose coefficients are polynomials in z etc. For example $y \cdot z \cdot x^2 + (z-1) \cdot x \cdot y^2 + 5$ is represented by $[5, \overline{[0, 0, \overline{[-1, 1]}\ z]}\ y, \overline{[0, \overline{[0, 1]}\ z]}\ y]\ x$.

The algorithm in §4.5.1 massively uses derivatives, the intermediate value theorem and the mean value theorem. For the qep. it is important to *compute* the derivative $p\prime$ of a

$$\begin{aligned}
\mathsf{datatype}\ \boldsymbol{\sigma} &= \mathsf{Z} \mid \mathsf{P} \mid \mathsf{N} \mid \not\mathsf{Z} \mid \top_{\boldsymbol{\sigma}} \\
S \models_{\boldsymbol{\sigma}} p : \mathsf{Z} &= \forall x \in S.p\ x = 0 \\
S \models_{\boldsymbol{\sigma}} p : \mathsf{P} &= \forall x \in S.p\ x > 0 \\
S \models_{\boldsymbol{\sigma}} p : \mathsf{N} &= \forall x \in S.p\ x < 0 \\
S \models_{\boldsymbol{\sigma}} p : \not\mathsf{Z} &= \forall x \in S.p\ x \neq 0 \\
S \models_{\boldsymbol{\sigma}} p : \top_{\boldsymbol{\sigma}} &= \forall x \in S.p\ x = p\ x
\end{aligned}$$

Figure 4.9: Signs and their interpretation

polynomial p, which is again a polynomial.

$$\begin{aligned}
\mathsf{pderivh}\ n\ [] = [] \quad \mid\ \mathsf{pderivh}\ n\ (c \cdot cs) &= (\underline{n} \cdot c) \cdot \mathsf{pderivh}\ (n+1)\ cs \\
p\prime &= \textbf{if}\ p = []\ \textbf{then}\ []\ \textbf{else}\ \mathsf{pderivh}\ 1\ p
\end{aligned}$$

In (4.58) we prove the main property of the syntactical derivative for polynomials: it represents the derivative of the polynomial function, indeed. In Isabelle, the derivative of a real function f at x is formalised by a predicate true for d when $\lim_{h \to 0} \frac{f(x+h)-f(x)}{h} = d$. Since derivatives are unique, the following notation using equality is not misleading.

$$\frac{d\overline{p}}{dx} = \overline{p\prime}\ x \tag{4.58}$$

The differentiability of polynomials is a trivial consequence of (4.58). The intermediate value theorem (4.59) and the mean value theorem (4.60) for polynomials follow hence from their general forms for continuous and differentiable functions. Moreover, the syntactical derivative is well-defined (cf. (4.61)) in the following sense: if two polynomials are equal (as functions) then so are the functions induced by their derivatives.

$$a < b \wedge \overline{p}\ a \cdot \overline{p}\ b < 0 \rightarrow \exists x.a < x < b \wedge \overline{p}\ x = 0 \tag{4.59}$$

$$a < b \rightarrow \exists x.a < x < b \wedge (\overline{p}\ b - \overline{p}\ a = (b-a) \cdot (\overline{p\prime}\ x)) \tag{4.60}$$

$$\overline{p} = \overline{q} \rightarrow \overline{p\prime} = \overline{q\prime} \tag{4.61}$$

Sign environments and matrices

Since the algorithm in §4.5.1 highly depends on signs and SM, we must represent them efficiently. Figure 4.9 shows the definition of a datatype $\boldsymbol{\sigma}$ for signs. The signs $\not\mathsf{Z}$ (for not zero) and $\top_{\boldsymbol{\sigma}}$ (for unknown) are useful to make formal statements about some steps of the qep. in §4.5.2, where some signs have not been *precisely* inferred yet. A sign s is precise ($\mathsf{precise}\ s$) if $s \in \{\mathsf{Z}, \mathsf{N}, \mathsf{P}\}$. In Figure 4.9, $S \models_{\boldsymbol{\sigma}} f : s$ formalises that the real function f has the sign s over the set S. Ultimately we lift this definition to a list fs of functions having respective signs ss over a set S. We formalise this by $S \models_{[\boldsymbol{\sigma}]} fs : ss$ below, which uses $\forall^2_{[]}$. Note that $\forall^2_{[]}\ P\ [x_1, \ldots, x_n]\ [y_1, \ldots, y_m]$ formalises $\bigwedge_{i=1}^{n} P\ x_i\ y_i$ for $n = m$ and *False* otherwise.

$$\begin{aligned}
\forall^2_{[]}\ P\ []\ [] = \mathit{True} \quad \mid\ \forall^2_{[]}\ P\ (x \cdot xs)\ [] &= \mathit{False} \quad \mid\ \forall^2_{[]}\ P\ []\ (x \cdot xs) = \mathit{False} \\
\forall^2_{[]}\ P\ (x \cdot xs)\ (y \cdot ys) &= P\ x\ y \wedge \forall^2_{[]}\ P\ xs\ ys \\
S \models_{[\boldsymbol{\sigma}]} fs : ss &= \forall^2_{[]}\ (\lambda f, s.S \models_{\boldsymbol{\sigma}} f : s)\ fs\ ss
\end{aligned}$$

In order to express the signs of real functions over a partition of the real line, we first introduce a function $\mathsf{partition}_{\mathbb{R}}$ to partition $\mathbb{R}$ into disjoint intervals and points. The result of $\mathsf{partition}_{\mathbb{R}}\ [a, b]$ is for instance $[] - \infty, a[, \{a\},]a, b[, \{b\},]b, +\infty[]$. For this to make sense we also define a list xs of real numbers to contain elements in strictly increasing order if $\mathsf{ordered}_{\mathbb{R}}\ xs$ holds. Note that $\mathsf{ordered}_{\mathbb{R}}\ [x_1, \ldots, x_n]$ formalises $\bigwedge_{i=1}^{n-1} x_i < x_{i+1}$.

$$\begin{aligned}\mathsf{partition}_{\mathbb{R}}\ [] = [] - \infty, +\infty[] \quad &| \ \mathsf{partition}_{\mathbb{R}}\ [x] = [] - \infty, x[, \{x\},]x, +\infty[] \\ | \ \mathsf{partition}_{\mathbb{R}}\ (x \cdot y \cdot ys) &= ([] - \infty, x[, \{x\},]x, y[])@tl\ (\mathsf{partition}_{\mathbb{R}}\ (y \cdot ys)) \\ \mathsf{ordered}_{\mathbb{R}}\ [] = \mathit{True} \quad &| \ \mathsf{ordered}_{\mathbb{R}}\ [x] = \mathit{True} \\ | \ \mathsf{ordered}_{\mathbb{R}}\ (x \cdot y \cdot ys) &= x < y \wedge \mathsf{ordered}_{\mathbb{R}}\ (y \cdot ys)\end{aligned}$$

Finally we formalise the SM for several functions over a partition of $\mathbb{R}$.

$$xs \models_{[[\boldsymbol{\sigma}]]} fs : m = \mathsf{ordered}_{\mathbb{R}}\ xs \wedge \forall^2_{[]}\ (\lambda S, ss.S \models_{[\boldsymbol{\sigma}]} fs : ss)\ (\mathsf{partition}_{\mathbb{R}}\ xs)\ m$$

Let $p_1 = \overline{[0,0,1]} = \lambda x.x^2$ and $p_2 = \overline{[-1,1]} = \lambda x.x - 1$ and x_1 and x_2 their respective roots. For instance $[x_1, x_2] \models_{[[\boldsymbol{\sigma}]]} [p_1, p_2] : [[\mathsf{P}, \mathsf{N}], [\mathsf{Z}, \mathsf{N}], [\mathsf{P}, \mathsf{N}], [\mathsf{P}, \mathsf{Z}], [\mathsf{P}, \mathsf{P}]]$ formalises

$$\begin{aligned}x_1 < x_2 \wedge (\forall x < x_1.p_1(x) > 0) \wedge (\forall x < x_1.p_2(x) < 0) \wedge p_1(x_1) = 0 \wedge p_2(x_1) < 0 \\ \wedge(\forall x > x_1.x < x_2 \rightarrow p_1(x) > 0) \wedge (\forall x > x_1.x < x_2 \rightarrow p_2(x) < 0) \\ \wedge p_1(x_2) > 0 \wedge p_2(x_2) = 0 \wedge (\forall x > x_2.p_1(x) > 0) \wedge (\forall x > x_2.p_2(x) > 0)\end{aligned}$$

Limit behaviour of real polynomials

After the formalisation of SE and SM, we formalise the relationship between the sign of a non-constant polynomials p and its derivative $p\prime$ near infinity. This helps us to formalise the step, in §4.5.1, of adding two "points" for $+\infty$ and $-\infty$ with signs determined by those of the derivative.

We define p to be non-constant if it has more than one coefficient and a non-zero head, i.e. $\mathsf{nonconstant}\ p$ in (4.62) holds. A key observation is that the degree of a non-constant polynomial p is even exactly when the degree of $p\prime$ is odd, cf. (4.63).

$$\mathsf{nonconstant}\ p = |p| > 1 \wedge \mathsf{last}\ p \neq 0 \tag{4.62}$$

$$\mathsf{nonconstant}\ p \rightarrow \mathsf{even}(\mathsf{deg}\ p) \leftrightarrow \mathsf{odd}\ (\mathsf{deg}\ p\prime) \tag{4.63}$$

Since the sign of p near infinity is determined by its monomial with highest degree, p and p' have the same behaviour near $+\infty$, cf. (4.66) and (4.67), and opposite signs near $-\infty$, cf. (4.65) and (4.64). The proofs involve monotonic and unbounded sequences and the mean-value theorem (4.60).

$$\mathsf{nonconstant}\ p \wedge (\forall x < l'.\overline{p\prime}\ x < 0) \rightarrow \exists l < l'.\forall y < l.\overline{p}\ y > 0 \tag{4.64}$$

$$\mathsf{nonconstant}\ p \wedge (\forall x < l'.\overline{p\prime}\ x > 0) \rightarrow \exists l < l'.\forall y < l.\overline{p}\ y < 0 \tag{4.65}$$

$$\mathsf{nonconstant}\ p \wedge (\forall x > u'.\overline{p\prime}\ x > 0) \rightarrow \exists u > u'.\forall y > u.\overline{p}\ y > 0 \tag{4.66}$$

$$\mathsf{nonconstant}\ p \wedge (\forall x > u'.\overline{p\prime}\ x < 0) \rightarrow \exists u > u'.\forall y > u.\overline{p}\ y < 0 \tag{4.67}$$

Sign behaviour of real polynomials

Using (4.64)–(4.67) we prove a formalisation of extending the SM by two "points" $+\infty$ and $-\infty$, cf. (4.68) and (4.69). Let us consider (4.68) more closely, (4.69) is analogous. Let p be non-constant and assume we have a SM for $p\prime$ and other polynomials, and we are in the process of inferring p's sign near $-\infty$. Property (4.68) considers a "snapshot" of the SM: the sign of p near $-\infty$ (the first interval) is not known yet (i.e. $\top_{\boldsymbol{\sigma}}$), and $p\prime$ is positive (i.e. P). Then (4.68) allows us to introduce a new point $z_{-\infty}$ and copy all entries for the other polynomials and set the sign of p on the new "point" and interval to negative (i.e. N). There are analogous versions, where $p\prime$ is negative.

$$\begin{aligned}
&\mathsf{nonconstant}\ p \wedge zs \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot ps) : ((\top_{\boldsymbol{\sigma}} \cdot \mathsf{P} \cdot r_1) \cdot m) \\
\rightarrow\ &\exists z_{-\infty}.(z_{-\infty} \cdot zs) \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot ps) : ((\mathsf{N} \cdot \mathsf{P} \cdot r_1) \cdot (\mathsf{N} \cdot \mathsf{P} \cdot r_1) \cdot (\top_{\boldsymbol{\sigma}} \cdot \mathsf{P} \cdot r_1) \cdot m) \qquad (4.68) \\
&\mathsf{nonconstant}\ p \wedge zs \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot ps) : (m@[a, b, \top_{\boldsymbol{\sigma}} \cdot \mathsf{P} \cdot r_2]) \\
\rightarrow\ &\exists z_{+\infty}.(zs@[z_{+\infty}]) \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot ps) : (m@[a, b, \top_{\boldsymbol{\sigma}} \cdot \mathsf{P} \cdot r_2, \mathsf{P} \cdot \mathsf{P} \cdot r_2, \mathsf{P} \cdot \mathsf{P} \cdot r_2]) \qquad (4.69)
\end{aligned}$$

For the sign inference over an interval with finite endpoints, we consider a corresponding "snapshot", cf. (4.70), and prove that we can replace the unknown sign for p by a corresponding one (P in (4.70)). We need all variants of (4.70), considering all the combinations of signs of p on the neighbour intervals and the sign of $p\prime$. The proofs use the intermediate value theorem (4.59).

$$\begin{aligned}
&(zs@x \cdot y \cdot zs') \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot fs) : (m@(\mathsf{P} \cdot r_1) \cdot (\top_{\boldsymbol{\sigma}} \cdot \mathsf{P} \cdot r_2) \cdot (\mathsf{P} \cdot r_3) \cdot m') \\
&\qquad \wedge |m| = 2 \cdot |zs| + 1 \\
\rightarrow\ &(zs@x \cdot y \cdot zs') \models_{[[\boldsymbol{\sigma}]]} (\overline{p} \cdot \overline{p\prime} \cdot fs) : (m@(\mathsf{P} \cdot r_1) \cdot (\mathsf{P} \cdot \mathsf{P} \cdot r_2) \cdot (\mathsf{P} \cdot r_3) \cdot m')
\end{aligned}$$

The sign inference on points is analogous. In (4.70), we consider again a "snapshot" of the SM at the polynomials we are interested in. Recall from §4.5.1 that at this stage we infer the sign of p at a point, which is a zero of q, using the sign of the pseud-remainder r of p by q, i.e. we have $a^n \cdot p(x) = q(x) \cdot s(x) + r(x)$, for some n and $s(x)$. We show a particular case in (4.70), but we need all the remaining combinations of signs.

$$\begin{aligned}
&zs@(x \cdot zs') \models_{[[\boldsymbol{\sigma}]]} (p \cdot ps@q \cdot qs@r \cdot rs) : (m@((\top_{\boldsymbol{\sigma}} \cdot s_1@\mathsf{Z} \cdot s_2@\mathsf{P} \cdot s_3) \cdot m')) \\
\wedge |ps| = |s_1| \wedge |qs| = |s_2| \wedge \mathsf{odd}\ |m| \wedge (\forall x.a^n \cdot p(x) = s(x) \cdot q(x) + r(x)) \wedge a \neq 0 \wedge \mathsf{even}\ n \\
&\qquad \rightarrow zs@x \cdot zs' \models_{[[\boldsymbol{\sigma}]]} (p \cdot ps@q \cdot qs@r \cdot rs) : (m@(\mathsf{P} \cdot s_1@\mathsf{Z} \cdot s_2@\mathsf{P} \cdot s_3) \cdot m')
\end{aligned} \qquad (4.70)$$

Sign inference is the main part of the procedure. From the 6000 lines of proofs we needed for *all* theorems involved in the procedure, we needed 5000 lines for limit and sign behaviour theorems.

Implementation

We ported the derived rule from HOL Light, presented in [MH05], which follows the implementation in [Har08, §5.9]. In §4.5.3, we present a formalisation of [Har08, §5.9] in detail. The computation and proofs about SM is organised by continuations, as we shall see in §4.5.3. Our implementation is at the debugging stage by now, but works for some examples. Debugging continuation passing style implementations is tedious. The implementation of the procedure itself is not as challenging as the proofs, needed to argue all the steps in §4.5.1. In [MH05] the authors share this experience. In our case we needed 6000 lines of Isabelle proofs, and 3500 lines of SML code.

4.5.3 An unfinished reflection

Unfortunately we have not been able to finish a reflection of §4.5.1 on time. Nevertheless, we think that there are some lessons and new problems worth presenting. We reflect RCF formulae by $\boldsymbol{\phi}$ and their semantics $(\!|.|\!)^{\cdot}$ in Figure 4.10. The type $\boldsymbol{\rho}$ is that of reflected multivariate polynomials in §3.1.2 but with rational coefficients. We present and motivate this change in more detail in the following. After that we present how we deal with SE and SM and how we transfer all the useful results of §4.5.2 to $\boldsymbol{\rho}$-polynomials. The rest of the section formalises a working qep. for RCF, presents our steps so far and raises the problems we encountered. For the formalisation of this reflection, we needed 1000 lines of Isabelle definitions for the implementation inside HOL. The presented reflection needed 4000 lines of Isabelle proofs so far.

$$\begin{array}{l}\mathsf{datatype}\ \phi \quad = \boldsymbol{T} \mid \boldsymbol{F} \mid \rho = \rho \mid \rho \neq \rho \mid \rho < \rho \mid \rho \leq \rho \mid \rho > \rho \mid \rho \geq \rho \\ \qquad\qquad \mid \neg\ \phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \longrightarrow \phi \mid \phi \leftrightarrow \phi \mid \exists\, \phi \mid \forall\, \phi\end{array}$$

$$\begin{array}{llllllllll}
(\!|\boldsymbol{T}|\!)^e & = & \mathit{True} & (\!|s \geq t|\!)^e & = & (\!|s|\!)^e_{\boldsymbol{\rho}} \geq (\!|t|\!)^e_{\boldsymbol{\rho}} & (\!|p \vee q|\!)^e & = & (\!|p|\!)^e \vee (\!|q|\!)^e \\
(\!|\boldsymbol{F}|\!)^e & = & \mathit{False} & (\!|s = t|\!)^e & = & ((\!|s|\!)^e_{\boldsymbol{\rho}} = (\!|t|\!)^e_{\boldsymbol{\rho}}) & (\!|p \longrightarrow q|\!)^e & = & (\!|p|\!)^e \rightarrow (\!|q|\!)^e \\
(\!|s < t|\!)^e & = & (\!|s|\!)^e_{\boldsymbol{\rho}} < (\!|t|\!)^e_{\boldsymbol{\rho}} & (\!|s \neq t|\!)^e & = & (\!|s|\!)^e_{\boldsymbol{\rho}} \neq (\!|t|\!)^e_{\boldsymbol{\rho}} & (\!|p \leftrightarrow q|\!)^e & = & (\!|p|\!)^e \leftrightarrow (\!|q|\!)^e \\
(\!|s > t|\!)^e & = & (\!|s|\!)^e_{\boldsymbol{\rho}} > (\!|t|\!)^e_{\boldsymbol{\rho}} & (\!|\neg p|\!)^e & = & \neg (\!|p|\!)^e & (\!|\exists\, p|\!)^e & = & \exists x.(\!|p|\!)^{x\cdot e} \\
(\!|s \leq t|\!)^e & = & (\!|s|\!)^e_{\boldsymbol{\rho}} \leq (\!|t|\!)^e_{\boldsymbol{\rho}} & (\!|p \wedge q|\!)^e & = & (\!|p|\!)^e \wedge (\!|q|\!)^e & (\!|\forall\, p|\!)^e & = & \forall x.(\!|p|\!)^{x\cdot e}
\end{array}$$

Figure 4.10: Syntax and semantics of RCF formulae

Polynomials

We use the $\boldsymbol{\rho}$-polynomials of §3.1.2 instantiated to the reals, but with a minor change: coefficients are rational, i.e. $\widehat{a : b}$ where $a :: \mathbb{Z}$ and $b :: \mathbb{Z}$. This modification is not necessary but it is more efficient and even simplifies some proofs. Assume for example that $2 \cdot x^2 - 3 \cdot y > 0$ is a sign assumption. When we encounter $-4 \cdot x^2 + 6 \cdot y$ we would like to deduce that it is negative. For this we make all polynomials in sign assumptions *monic*, i.e. we divide them by their head's coefficient. To achieve the previous goal without using rational numbers, we would need e.g. complicated gcd computations. Rational numbers are just pairs of integers $i : j$ interpreted as field elements by $(\!|i : j|\!)_{\boldsymbol{r}} = \underline{i}/\underline{j}$. This representation is not unique in general, but it is for those in normal form (i.e. satisfying $\mathsf{isnorm}_{\boldsymbol{r}}$), see (4.71):

$$\begin{array}{r}\mathsf{isnorm}_{\boldsymbol{r}}\ (i : j) = \textbf{if}\ i = 0\ \textbf{then}\ j = 0\ \textbf{else}\ b > 0 \wedge \mathit{gcd}\ i\ j = 1 \\ \mathsf{isnorm}_{\boldsymbol{r}}\ a \wedge \mathsf{isnorm}_{\boldsymbol{r}}\ b \rightarrow (\!|a|\!)_{\boldsymbol{r}} = (\!|b|\!)_{\boldsymbol{r}} \leftrightarrow a = b \end{array} \tag{4.71}$$

We implement the usual arithmetical operations $(+_{\boldsymbol{r}}, \cdot_{\boldsymbol{r}}, -_{\boldsymbol{r}}, /_{\boldsymbol{r}})$ and ordering relations $(<_{\boldsymbol{r}}, \leq_{\boldsymbol{r}})$ and prove them correct:

$$\begin{array}{r}\mathsf{isnorm}_{\boldsymbol{r}}\ a \wedge \mathsf{isnorm}_{\boldsymbol{r}}\ b \rightarrow \\ (\!|a \circ b|\!)_{\boldsymbol{r}} = (\!|a|\!)_{\boldsymbol{r}} \circ (\!|b|\!)_{\boldsymbol{r}} \wedge (a \bowtie b \leftrightarrow (\!|a|\!)_{\boldsymbol{r}} \bowtie (\!|b|\!)_{\boldsymbol{r}}) \wedge \mathsf{isnorm}_{\boldsymbol{r}}\ (a \circ b),\end{array} \tag{4.72}$$

for $(\circ, \circ) \in \{(+_{\boldsymbol{r}}, +), (\cdot_{\boldsymbol{r}}, \cdot), (-_{\boldsymbol{r}}, -), (/_{\boldsymbol{r}}, /)\}$ and $(\bowtie, \bowtie) \in \{(<_{\boldsymbol{r}}, <), (\leq_{\boldsymbol{r}}, \leq)\}$.

It is important to note that the implementation does not raise an exception but correctly returns 0 as a result for $x/0$. This reflects Isabelle/HOL's behaviour correctly. We also provide a function $\mathsf{norm}_{\boldsymbol{r}}$ to normalise any $i : j$ and prove it correct. In the following $\boldsymbol{i}_{\boldsymbol{r}}$ denotes $i : 1$ for any integer $i \neq 0$ and $\boldsymbol{0}_{\boldsymbol{r}}$ denotes $0 : 0$. We adapt the normal form ($\mathsf{ishornh}$ in §3.1.2) to enforce that the rational coefficients are in normal form. *All theorems we presented in §3.1.2 still hold.*

We implement the derivative of $p :: \boldsymbol{\rho}$ with respect to the bound variable $\boldsymbol{v}_0$ by $\frac{dp}{d\boldsymbol{v}_0}$.

$$\begin{array}{r}\mathsf{pderivh}_{\boldsymbol{\rho}}\ n\ p = \textbf{case}\ p\ \textbf{of}\ c + \boldsymbol{v}_0 * p\ \Rightarrow (\widehat{\boldsymbol{n}_{\boldsymbol{r}}} \circledast c) + \boldsymbol{v}_0 * (\mathsf{pderivh}_{\boldsymbol{\rho}}\ (n+1)\ p) \\ \mid\ \_\ \Rightarrow \widehat{\boldsymbol{n}_{\boldsymbol{r}}} \circledast p\end{array}$$

$$\frac{dp}{d\boldsymbol{v}_0} = \textbf{case}\ p\ \textbf{of}\ c + \boldsymbol{v}_0 * q\ \Rightarrow \mathsf{pderivh}_{\boldsymbol{\rho}}\ 1\ p\ \mid\ \_\ \Rightarrow \frac{dp}{d\boldsymbol{v}_0} = \boldsymbol{0}_{\boldsymbol{\rho}}$$

Recall from §3.1.2 that given an environment $e :: [\mathbb{R}]$ we can view $p :: \boldsymbol{\rho}$ as *univariate* in the sense of §3.1.1 by considering $[p]^e$. Recall also the key property (3.26). Ultimately we relate both derivative implementations:

$$\mathsf{ishorn}\ p \rightarrow (\!|\frac{dp}{d\boldsymbol{v}_0}|\!)^{x\cdot e}_{\boldsymbol{\rho}} = \overline{([p]^{x\cdot e})'}\ x \tag{4.73}$$

The proof of (4.73) is by induction on p. Using this technique ("changing the point of view") we transfer all the tediously proved theorems in §4.5.2 to $\boldsymbol{\rho}$-polynomials. Subsequently we define interpretation of signs for $\boldsymbol{\rho}$-polynomials such that all the signs-related theorems of §4.5.2 are transferred to $\boldsymbol{\rho}$-polynomials.

Sign environments and matrices

In contrast to §4.5.2, we need SE and SM only for computations. More precisely, we need the interpretation of SE and SM (e.g. $xs \models_{[[\boldsymbol{\sigma}]]} fs : m$ in §4.5.2) *only* to prove our implementation correct. The first step is to interpret signs for $\boldsymbol{\rho}$-polynomials and to establish a connection to all useful theorems proved in §4.5.2. We use the "changing point of view" to define the interpretations:

$$S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s = S \models_{\boldsymbol{\sigma}} [p]^e : s \tag{4.74}$$

$$S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} ps : ss = \forall^2_{[]}\ (\lambda p, s. S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s)\ ps\ ss \tag{4.75}$$

$$xs, e \models^{\boldsymbol{\rho}}_{[[\boldsymbol{\sigma}]]} ps : m = \mathsf{ordered}_{\mathbb{R}}\ xs \wedge \forall^2_{[]}\ (\lambda S, ss. S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} ps : ss)\ (\mathsf{partition}_{\mathbb{R}}\ xs)\ m \tag{4.76}$$

In the implementation we represent a SE by an association list $se :: [\boldsymbol{\rho} \times \boldsymbol{\sigma}]$. An association list se describes a partial function (denoted by $\overrightarrow{se} :: \boldsymbol{\rho} \rightharpoonup \boldsymbol{\sigma}$) mapping p to $\lfloor s \rfloor$, if (p, s) is the first occurrence in se, and $\bot$ otherwise. Here $\lfloor\rfloor$ and $\bot$ are the constructors of the option type $\lfloor\alpha\rfloor$ over the HOL type α. The HOL type $\alpha \rightharpoonup \beta$ is just a suggestive notation for $\alpha \Rightarrow \lfloor\beta\rfloor$. We use $\mathsf{del}\ p\ se$ to remove all entries for p in se. In the following we often need to lift functions to deal with option-values. For this we define $\ulcorner f \urcorner$ and $\ulcorner g_\bot \urcorner$ below. For a binary function g we also denote its lifted version by $\ulcorner g \urcorner$. Note the strictness with respect to $\bot$. Since the implementation in [Har08] heavily uses exceptions, the lifters below were very helpful for an implementation in HOL. In most cases, an exception in [Har08] means that the we have proved *False* from the current SM. We deal with this using $\ulcorner f_{\boldsymbol{F}} \urcorner$ to lift a function $f :: \alpha \Rightarrow \boldsymbol{\phi}$ to deal with option-values where $\bot$ is interpreted as $\boldsymbol{F}$.

$$\begin{aligned}
\ulcorner f \urcorner\ x &= \mathbf{case}\ x\ \mathbf{of}\ \bot \Rightarrow \bot \mid \lfloor y \rfloor \Rightarrow \lfloor f\ y \rfloor \\
\ulcorner g_\bot \urcorner\ x &= \mathbf{case}\ x\ \mathbf{of}\ \bot \Rightarrow \bot \mid \lfloor y \rfloor \Rightarrow g\ y \\
\ulcorner f_{\boldsymbol{F}} \urcorner\ x &= \mathbf{case}\ x\ \mathbf{of}\ \bot \Rightarrow \boldsymbol{F} \mid \lfloor y \rfloor \Rightarrow f\ y \\
\ulcorner g \urcorner\ x\ y &= \mathbf{case}\ (x, y)\ \mathbf{of}\ (\lfloor a \rfloor, \lfloor b \rfloor) \Rightarrow \lfloor g\ a\ b \rfloor \mid _ \Rightarrow \bot
\end{aligned}$$

Since se contains both polynomials and signs we also write $S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se$ to denote $\forall (p, s) \in \{\!\{se\}\!\}. S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s$ or equivalently $S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} (\mathsf{map}\ fst\ se) : (\mathsf{map}\ snd\ se)$. We use and maintain *well-formed* SE satisfying wf: no polynomial is assigned a sign twice and all signs are precise.

$$\mathsf{wf}\ se = \mathsf{distinct}\ (\mathsf{map}\ fst\ se) \wedge \forall (p, s) \in \{\!\{se\}\!\}. \mathsf{precise}\ s$$

In SE, we also ensure that all polynomials are normalised and *monic*, i.e. their leading coefficient is $\mathbf{1}_{\boldsymbol{r}}$. For that we define $\mathsf{monic}\ p$ to return a pair (q, b), where q is the resulting monic polynomial and b is *True* if and only if the leading coefficient of p was negative. This is important to know since we must swap the sign under consideration if we divide by a negative number. This swapping is done by $\mathsf{swap}_{\boldsymbol{\sigma}}$.

$$\begin{aligned}
&\mathsf{swap}_{\boldsymbol{\sigma}}\ s\ b && = \mathbf{if}\ \neg\ b\ \mathbf{then}\ s\ \mathbf{else}\ (\mathbf{case}\ s\ \mathbf{of}\ \mathsf{P} \Rightarrow\ \mathsf{N} \mid \mathsf{N} \Rightarrow\ \mathsf{P} \mid _ \Rightarrow\ s) \\
&\mathsf{hconst}\ (c + \boldsymbol{v}_n * p) && = \mathsf{hconst}\ p \\
&\mathsf{hconst}\ \widehat{a} && = a \\
&\mathsf{monic}\ p && = \mathbf{let}\ h = \mathsf{hconst}\ p \\
& && \quad \mathbf{in\ if}\ h = \mathbf{0}_{\boldsymbol{r}}\ \mathbf{then}\ (p, \mathit{False})\ \mathbf{else}\ (\widehat{\mathbf{1}_{\boldsymbol{r}} /_{\boldsymbol{r}} h} \circledast p, h <_{\boldsymbol{r}} \mathbf{0}_{\boldsymbol{r}})
\end{aligned}$$

For monic we prove two key properties: (4.77) states that monic divides by the head's coefficient, and (4.78) relates the sign of a polynomial and that of its monic version.

$$\mathsf{ishorn}\ p \rightarrow (\!|\mathsf{hconst}\ p|\!)^e_{\boldsymbol{\rho}} \cdot (\!|fst\ (\mathsf{monic}\ p)|\!)^e_{\boldsymbol{\rho}} = (\!|p|\!)^e_{\boldsymbol{\rho}} \tag{4.77}$$

$$\mathsf{monic}\ p = (q, b) \rightarrow S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} q : (\mathsf{swap}_{\boldsymbol{\sigma}}\ b\ s) = S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s \tag{4.78}$$

$$\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \mathbf{let}\ (q, b) = \mathsf{monic}\ p\ \mathbf{in}\ \ulcorner(\mathsf{swap}_{\boldsymbol{\sigma}}\ b)\urcorner\ (\overrightarrow{se}\ q)$$

$$\begin{array}{l} se[p \leftarrow s] = \\ \mathbf{if}\ p = \mathbf{0}_{\boldsymbol{\rho}}\mathbf{then\ if}\ s = \mathsf{Z}\ \mathbf{then}\ \lfloor se \rfloor \mathbf{else}\ \bot \\ \mathbf{else\ let}\ (p', b) = \mathsf{monic}\ p\ ;\ s' = \mathsf{swap}_{\boldsymbol{\sigma}}\ b\ s\ ; \\ t = (\mathbf{case}\ \overrightarrow{se}\ p'\ \mathbf{of}\ \lfloor t' \rfloor \Rightarrow t'\ \mid \bot \Rightarrow s') \\ \mathbf{in\ if}\ s' = t\ \mathbf{then}\ \lfloor (p', s') \cdot (\mathsf{del}\ p'\ se) \rfloor\ \mathbf{else}\ \bot \end{array}$$

Figure 4.11: Finding and asserting signs

In Figure 4.11 we define two main operations on a SE se. Given $p :: \boldsymbol{\rho}$, $\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p$ tries to find p's sign in se. The call $se[p \leftarrow s]$ tries to add the new assumption "p has sign s" to se. Both functions consider monic polynomials and both might fail (i.e. return $\bot$). Failure in $\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p$ occurs exactly when $q \notin \mathsf{dom}(\overrightarrow{se})$, where $\mathsf{monic}\ p = (q, b)$, cf. (4.79). In the success case $\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \lfloor s \rfloor$ we prove that $S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s$ is a consequence of $S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se$, for well-formed se, cf. (4.80).

$$\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \bot \leftrightarrow fst\ (\mathsf{monic}\ p) \notin \mathsf{dom}(\overrightarrow{se}) \tag{4.79}$$

$$\mathsf{ishorn}\ p \wedge \mathsf{wf}\ se \wedge \mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \lfloor s \rfloor \wedge S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se \rightarrow S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s \tag{4.80}$$

If sign assertion fails, $se[p \leftarrow s] = \bot$, then assuming p has sign s *contradicts* the remaining assumptions in se. If $se[p \leftarrow s] = \lfloor se_0 \rfloor$, then se_0 contains both, the old assumptions in se and the new one. Property (4.81) formalises this explanation in a stronger form (note the $\leftrightarrow$). Moreover, if se is well-formed, then so is se_0, cf. (4.82).

$$\begin{array}{r} S \neq \emptyset \wedge \mathsf{precise}\ s \wedge \mathsf{ishorn}\ p \wedge \mathsf{wf}\ se \rightarrow \\ (\mathbf{case}\ se[p \leftarrow s]\ \mathbf{of}\ \lfloor se_0 \rfloor \Rightarrow S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se_0\ \mid\ \bot \Rightarrow \mathit{False}) \leftrightarrow S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s \end{array} \tag{4.81}$$

$$se[p \leftarrow s] = \lfloor se_0 \rfloor \wedge \mathsf{wf}\ se \rightarrow \mathsf{wf}\ se_0 \tag{4.82}$$

Sign inference

The sign inference takes place in $\mathsf{deduce}_{\boldsymbol{\sigma}}$ (see Figure 4.12). Recall from §4.5.1 that to determine a SE for $p, p_1, \ldots, p_n$ we just need one for $p_0, p_1, \ldots, p_n, r_0, \ldots, r_n$, where $p_0 = p'$ and r_i is the pseudo-division remainder of p by p_i, for $0 \leq i \leq n$. In the call $\mathsf{deduce}_{\boldsymbol{\sigma}}\ ct\ m$, we assume that m is a SM environment for $p_0, \ldots, p_n, r_0, \ldots, r_n$, and that ct is the actual continuation. Note that $|hd\ m| = 2 \cdot (n + 1)$ and hence splitting the SE in m into the half (using chop), we obtain two separate SEs: pe for $p_0, \ldots, p_n$ and re for $r_0, \ldots, r_n$. Ultimately we infer the signs of p at the points where one of the p_i's is zero, using $\mathsf{infer}_{\boldsymbol{\sigma}}\ pe\ re$. Now we can throw away all points (and their neighbouring intervals) where we have no zeros (using red). This yields the temporary SM m_1. In the next step, $\mathsf{deduce}_{\boldsymbol{\sigma}}$ extends m_1 with two "points" for $-\infty$ and $+\infty$ (this is m_2), whose respective signs are determined by the sign of the derivative $p_0 = p'$ at the first and last point, respectively. For $-\infty$ we must swap the sign, since near $-\infty$, p and p' have opposite signs, due to the even vs. odd behaviour of p's and p''s degrees. Ultimately the step of inferring the right signs on the intervals and determining the new zeros of p is done by $\mathsf{infer}_{]\boldsymbol{\sigma}[}\ m_2$. Of course we remove the two $\pm\infty$-"points" after this step. This is simply done by $\mathsf{butlast} \circ\ tl$, but since $\mathsf{infer}_{]\boldsymbol{\sigma}[}\ m_2$ might fail, we lift it. The result is m_3, which is the right SM for $p, p_0, \ldots, p_n$. In the last step we just remove all entries of p_0, reduce the matrix and finally apply the continuation. Note that $\mathsf{infer}_{]\boldsymbol{\sigma}[}\ m_2$ fails if and only if m_2 has inconsistent signs or a *point* with an imprecise sign. We have proved all "mathematical" theorems needed for sign inference in §4.5.2. We have not proved anything worth mentioning for the reflective implementation.

$$
\begin{array}{l}
\mathsf{infer}_{\boldsymbol{\sigma}}\ pe\ re = \textbf{case}\ \mathsf{index}\ (\lambda x.x = \mathsf{Z})\ pe\ \textbf{of}\ \lfloor i \rfloor \Rightarrow (re!i) \cdot pe\ \mid\ \bot\ \Rightarrow \not\mathsf{Z} \cdot pe \\
\mathsf{red}\ (is \cdot xs \cdot r) \ = \ \textbf{let}\ r' = \mathsf{red}\ r\ \textbf{in if}\ (\mathsf{Z} \in \{\!\!\{xs\}\!\!\})\ \textbf{then}\ is \cdot xs \cdot r'\ \textbf{else}\ r' \\
\mathsf{red}\ m \ = \ m \\
\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ((l \cdot ls) \cdot (i \cdot is) \cdot (r \cdot rs) \cdot m) \ = \\
\quad \textbf{if}\ (l = \mathsf{Z} \wedge r = \mathsf{Z}) \vee l = \not\mathsf{Z} \vee r = \not\mathsf{Z}\ \textbf{then}\ \bot \\
\quad \textbf{else if}\ l = \mathsf{Z}\ \textbf{then}\ \ulcorner(\lambda xs.(l \cdot ls) \cdot (r \cdot is) \cdot xs)\urcorner\ (\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ((r \cdot rs) \cdot m)) \\
\quad \textbf{else if}\ r = \mathsf{Z}\ \textbf{then}\ \ulcorner(\lambda xs.(l \cdot ls) \cdot (l \cdot is) \cdot xs)\urcorner\ (\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ((r \cdot rs) \cdot m)) \\
\quad \textbf{else if}\ l = r \wedge r \in \{\mathsf{N}, \mathsf{P}\} \\
\qquad \textbf{then}\ \ulcorner(\lambda xs.(l \cdot ls) \cdot (l \cdot is) \cdot xs)\urcorner\ (\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ((r \cdot rs) \cdot m)) \\
\qquad \textbf{else}\ \ulcorner(\lambda xs.(l \cdot ls) \cdot (l \cdot is) \cdot (\mathsf{Z} \cdot is) \cdot (r \cdot is) \cdot xs)\urcorner\ (\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ((r \cdot rs) \cdot m)) \\
\mathsf{infer}_{]\boldsymbol{\sigma}[}\ ps = \lfloor ps \rfloor
\end{array}
$$

$$
\begin{array}{l}
\mathsf{deduce}_{\boldsymbol{\sigma}}\ ct\ m\ = \\
\quad \textbf{let}\ l = \frac{\lfloor hd\ m \rfloor}{2}\ ; \\
\qquad m_1 = \mathsf{red}\ (\mathsf{map}((\lambda(x,y).\mathsf{infer}_{\boldsymbol{\sigma}}\ x\ y) \circ (\mathsf{chop}\ l))m)\ ; \\
\qquad m_2 = ([\mathsf{swap}_{\boldsymbol{\sigma}}\ True\ ((hd\ m_1)!1)] \cdot m_1)@[[(\mathsf{last}\ m_1)!1]]\ ; \\
\qquad m_3 = \ulcorner(\mathsf{butlast} \circ tl)\urcorner\ (\mathsf{infer}_{]\boldsymbol{\sigma}[}\ m_2) \\
\quad \textbf{in}\ \ulcorner ct_{\boldsymbol{F}} \urcorner\ (\ulcorner \mathsf{red} \urcorner\ (\ulcorner(\mathsf{map}\ (\lambda l.(hd\ l) \cdot (tl\ (tl\ l))))\urcorner m_3))
\end{array}
$$

Figure 4.12: Sign inference

$$
\begin{array}{ll}
\mathsf{test}_{\boldsymbol{\phi}}\ se\ (f \diamond g) & = \mathsf{test}_{\boldsymbol{\phi}}\ se\ f \diamond \mathsf{test}_{\boldsymbol{\phi}}\ se\ g \\
 & \qquad \text{for } (\diamond, \diamond) \in \{(\wedge,\wedge),(\vee,\vee),(\longrightarrow,\rightarrow),(\leftrightarrow,\leftrightarrow)\} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p = \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s = \mathsf{Z}\ \mid\ \_ \Rightarrow \mathit{False} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p \neq \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s \neq \mathsf{Z}\ \mid\ \_ \Rightarrow \mathit{False} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p < \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s = \mathsf{N}\ \mid\ \_ \Rightarrow \mathit{False} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p > \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s = \mathsf{P}\ \mid\ \_ \Rightarrow \mathit{False} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p \leq \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s \in \{\mathsf{Z}, \mathsf{N}\}\ \mid\ \_ \Rightarrow \mathit{False} \\
\mathsf{test}_{\boldsymbol{\phi}}\ (p \geq \mathbf{0}_{\boldsymbol{\rho}}) & = \textbf{case}\ \overrightarrow{se}\ p\ \textbf{of}\ \lfloor s \rfloor \Rightarrow s \in \{\mathsf{Z}, \mathsf{P}\}\ \mid\ \_ \Rightarrow \mathit{False}
\end{array}
$$

Figure 4.13: Evaluation of $\boldsymbol{\phi}$-formulae in SE

The main loop

Before we explain the main loop, let us consider $\mathsf{test}_{\boldsymbol{\phi}}$, the evaluation of a formula in a SE (see Figure 4.13). Let $\mathsf{pols}\ f$ (not shown) compute a list of all polynomials occurring in a qf. $\boldsymbol{\phi}$-formula f. For any well-formed SE se defining the signs of all polynomials in $\mathsf{pols}\ f$, f is semantically equivalent to the result of $\mathsf{test}_{\boldsymbol{\phi}}$ over S, whenever $S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se$ holds, cf. (4.83).

$$
\begin{array}{r}
\mathsf{qfree}\ f \wedge \mathsf{wf}\ se \wedge \{\!\!\{\mathsf{pols}\ f\}\!\!\} \subseteq \mathsf{dom}(\overrightarrow{se}) \wedge S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se \\
\rightarrow \forall x \in S.(\!|f|\!)^{x \cdot e} \leftrightarrow \mathsf{test}_{\boldsymbol{\phi}}\ se\ f
\end{array}
\tag{4.83}
$$

Given a qf. $\boldsymbol{\phi}$-formula f, the qep. for one $\exists$ (see $\mathsf{qe}_{\exists\mathbb{R}}$ in Figure 4.14) first simplifies f to g and then computes $ps = \mathsf{pols}\ g$ and ultimately calls the main loop (see casesplit in Figure 4.15). Before we continue, we must clarify the meaning of Figure 4.15. It represents what we *input* to Isabelle. Since we have not proved the termination of these functions, *the equations in* Figure 4.15 *do not hold in Isabelle/HOL. This is the unique exception in this thesis.* The main loop is organised in three mutually recursive functions, all of them are continuation-based. In our context we have two kinds of continuations: one for SE of type $[\boldsymbol{\rho} \times \boldsymbol{\sigma}] \Rightarrow \boldsymbol{\phi}$, and one for SM of type $[[\boldsymbol{\sigma}]] \Rightarrow \boldsymbol{\phi}$. We use SE-continuations to split over the

$$
\begin{array}{lcl}
\mathsf{qe}_{\exists\mathbb{R}}\ f & = & \textbf{let}\ g = \mathsf{simp}_{\boldsymbol{\phi}}\ f\ ;\ ps = \mathsf{pols}\ g\ ; \\
 & & \quad ct = (\lambda m.\textbf{if}\ (\exists se \in \{\!\{m\}\!\}.\mathsf{test}_{\boldsymbol{\phi}}\ g\ (\mathsf{zip}\ ps\ se))\ \textbf{then}\ \boldsymbol{T}\ \textbf{else}\ \boldsymbol{F}) \\
 & & \textbf{in}\ (\mathsf{simp}_{\boldsymbol{\phi}} \circ \mathsf{decr}_{\boldsymbol{\phi}})\ (\mathsf{casesplit}\ []\ ps\ ct\ [(\mathbf{0}_{\boldsymbol{\rho}}, \mathsf{Z}), (\mathbf{1}_{\boldsymbol{\rho}}, \mathsf{P})]) \\
\mathsf{qe}_{\mathbb{R}} & = & \mathsf{qelim}\ \mathsf{qe}_{\exists\mathbb{R}}
\end{array}
$$

Figure 4.14: The main qe. procedure

$$
\begin{array}{l}
\mathsf{casesplit}\ ds\ []\ ct\ se = \mathsf{matrix}\ ds\ ct\ se \\
\mathsf{casesplit}\ ds\ (p \cdot ps)\ ct\ se = \\
\quad \mathsf{split}_{\boldsymbol{\sigma}}\ (\mathsf{head}\ 0\ p) \\
\qquad (\textbf{if}\ \mathsf{unbound}_{\boldsymbol{\rho}}\ p\ \textbf{then}\ \mathsf{delconst}\ ds\ p\ ps\ ct\ \textbf{else}\ \mathsf{casesplit}\ ds\ (\mathsf{behead}\ p \cdot ps)\ ct) \\
\qquad (\textbf{if}\ \mathsf{unbound}_{\boldsymbol{\rho}}\ p\ \textbf{then}\ \mathsf{delconst}\ ds\ p\ ps\ ct\ \textbf{else}\ \mathsf{casesplit}\ (ds@[p])\ ps\ ct)\ se \\
\mathsf{delconst}\ ds\ p\ ps\ ct\ se = \\
\quad \mathsf{casesplit}\ ds\ ps\ (\lambda m.ct\ (\mathsf{map}\ (\mathsf{insertat}\ |ds|\ (the\ (\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p)))\ m))\ se \\
\mathsf{matrix}\ []\ ct\ se = ct\ [[]] \\
\mathsf{matrix}\ (a \cdot as)\ ct\ se = \\
\quad \textbf{let}\ ps = a \cdot as\ ;\ p = \mathsf{foldr}\ (\lambda p, q.\textbf{if}\ \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p > \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ q\ \textbf{then}\ p\ \textbf{else}\ q)\ as\ a \\
\qquad p' = \frac{dp}{d\boldsymbol{v}_0}; i = the\ (\mathsf{index}\ (\lambda x.x = p)\ ps)\ ; \\
\qquad qs = (\textbf{let}\ (p_1, p_2) = \mathsf{chop}\ i\ ps\ \textbf{in}\ p' \cdot p_1@tl\ p_2) \\
\qquad gs = \mathsf{foldr}\ ((\ulcorner(\lambda x, xs.x \cdot xs)\urcorner) \circ (\mathsf{pdiv}_{\boldsymbol{\rho}>0}\ se\ p))\ qs\ \lfloor[]\rfloor\ ; \\
\qquad ct' = (\lambda m.ct\ (\mathsf{map}\ (\lambda l.\mathsf{insertat}\ i\ (hd\ l)\ (tl\ l))\ m)) \\
\quad \textbf{in case}\ gs\ \textbf{of}\ \bot\ \Rightarrow \boldsymbol{F}\ |\ \lfloor gs' \rfloor \Rightarrow \mathsf{casesplit}\ []\ (qs@gs')\ (\mathsf{deduce}_{\boldsymbol{\sigma}}\ ct')\ se
\end{array}
$$

Figure 4.15: The main loop

heads of the polynomials in ps in casesplit. This splitting is organised by function $\mathsf{split}_{\boldsymbol{\sigma}}$, which we consider more closely later on. During case split we detect constant polynomials ($\mathsf{unbound}_{\boldsymbol{\rho}}$) and add their signs to the SM using delconst. Function casesplit also records which polynomials have been considered so far and moves them to ds (cf. Figure 4.15). We start the main loop with the SE for $\mathbf{0}_{\boldsymbol{\rho}}$ and $\mathbf{1}_{\boldsymbol{\rho}}$ only.

After splitting over all head's signs, we compute the SM of ps (by then copied into ds) using matrix. Function matrix computes the SM as we described it in §4.5.1. We use SM-continuations to express the recursive SM computations using derivative and pseudo-division (see Figure 4.16). Function $\mathsf{pdiv}_{\boldsymbol{\rho}>0}$ returns the pseudo-remainder possibly negated depending on the head's sign in the actual SE. for later computations, the sign of p will be exactly the sign of r, cf. (4.86). Note also that $\mathsf{pdiv}_{\boldsymbol{\rho}>0}$ fails exactly when the head's sign is zero or undefined, cf. (4.84), and that the degree of the result is *strictly* less than the input, cf. (4.85). This last property plays an important role in the termination proof of the main loop.

$$\mathsf{pdiv}_{\boldsymbol{\rho}>0}\ se\ p\ q = \bot \leftrightarrow (\mathsf{find}_{\boldsymbol{\sigma}}\ se\ (\mathsf{head}\ q\ 0) \in \{\bot, \lfloor \mathsf{Z} \rfloor\}) \tag{4.84}$$

$$\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \wedge \neg\mathsf{unbound}_{\boldsymbol{\rho}}\ q \wedge \mathsf{pdiv}_{\boldsymbol{\rho}>0}\ se\ p\ q = \lfloor r \rfloor \rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ r\ 0 < \mathsf{deg}_{\boldsymbol{\rho}}\ q\ 0 \tag{4.85}$$

$$\begin{array}{r}
\mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \wedge q \neq \mathbf{0}_{\boldsymbol{\rho}} \wedge \mathsf{wf}\ se \wedge \mathsf{pdiv}_{\boldsymbol{\rho}>0}\ se\ p\ q = \lfloor r \rfloor \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} q : \mathsf{Z} \\
\rightarrow \forall s.\mathsf{precise}\ s \rightarrow S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s \leftrightarrow S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} r : s
\end{array} \tag{4.86}$$

$$
\begin{array}{l}
\mathsf{pdiv}_{\boldsymbol{\rho}>0}\ se\ s\ p = \\
\textbf{let}\ a = \mathsf{head}\ 0\ p\ ;\ (k, r) = s \oslash p\ ;\ t = \mathsf{find}_{\boldsymbol{\sigma}}\ se\ a \\
\textbf{in if}\ t \in \{\lfloor \mathsf{Z} \rfloor, \bot\}\textbf{then}\ \bot \\
\quad \textbf{else if}\ (t = \lfloor \mathsf{P} \rfloor \vee \mathsf{even}\ k)\ \textbf{then}\ \lfloor r \rfloor\textbf{else}\ \lfloor \ominus r \rfloor
\end{array}
$$

Figure 4.16: Pseudo-division within SE

$$\begin{array}{l}\mathsf{split}_{\boldsymbol{\sigma}}\ p\ ct_{\mathsf{Z}}\ ct_{\not\mathsf{Z}}\ se = \textbf{case}\ \mathsf{find}_{\boldsymbol{\sigma}}\ se\ p\ \textbf{of}\\ \lfloor s\rfloor \Rightarrow (\textbf{if}\ s = \mathsf{Z}\ \textbf{then}\ ct_{\mathsf{Z}}\ \textbf{else}\ ct_{\not\mathsf{Z}})se\\ \mid\ \bot \Rightarrow p = \mathbf{0}_{\boldsymbol{\rho}} \wedge \ulcorner ct_{\mathsf{Z}\boldsymbol{F}}\urcorner\ (se[p \leftarrow \mathsf{Z}]) \vee\\ p > \mathbf{0}_{\boldsymbol{\rho}} \wedge \ulcorner ct_{\not\mathsf{Z}_{\boldsymbol{F}}}\urcorner\ (se[p \leftarrow \mathsf{P}]) \vee p < \mathbf{0}_{\boldsymbol{\rho}} \wedge \ulcorner ct_{\not\mathsf{Z}_{\boldsymbol{F}}}\urcorner (se[p \leftarrow \mathsf{N}])\end{array}$$

Figure 4.17: Organisation of case splittings on signs

We call the main loop with a simple SM-continuation ct (cf. also Figure 4.14)

$$ct = \lambda m.\textbf{if}\ (\exists se \in \{\!\!\{m\}\!\!\}.\mathsf{test}_{\boldsymbol{\phi}}\ g\ (\mathsf{zip}\ ps\ se))\ \textbf{then}\ \boldsymbol{T}\ \textbf{else}\ \boldsymbol{F}.$$

Given a SM m, we test if there is a SE satisfying g. Function matrix updates this basic SM-continuation by adding the signs computed so far, and finally calls it on the empty SM, when no polynomials need to be considered. Due to the updates on the SM-continuation, the full SM is computed out of the empty SM and all the case splits are encoded in the resulting formula. The final step of $\mathsf{qe}_{\exists\mathbb{R}}$ decreases the de Brujin indices and simplifies, cf. Figure 4.14.

Let us now consider $\mathsf{split}_{\boldsymbol{\sigma}} :: \boldsymbol{\rho} \Rightarrow ([\boldsymbol{\rho} \times \boldsymbol{\sigma}] \Rightarrow \boldsymbol{\phi}) \Rightarrow ([\boldsymbol{\rho} \times \boldsymbol{\sigma}] \Rightarrow \boldsymbol{\phi}) \Rightarrow [\boldsymbol{\rho} \times \boldsymbol{\sigma}] \Rightarrow \boldsymbol{\phi}$ in Figure 4.17, the organiser of case splittings. The result of $\mathsf{split}_{\boldsymbol{\sigma}}\ p\ ct_{\mathsf{Z}}\ ct_{\not\mathsf{Z}}\ se$ is a $\boldsymbol{\phi}$-formula where we split over the signs of p but taking the SE se into account. Further computations are performed by the two SE-continuations ct_{Z} and $ct_{\not\mathsf{Z}}$, which already assume that p has sign zero and not zero (i.e. positive or negative) respectively. Note that $\mathsf{split}_{\boldsymbol{\sigma}}\ p\ ct_{\mathsf{Z}}\ ct_{\not\mathsf{Z}}$ is itself a SE-continuation. Interesting properties about $\mathsf{split}_{\boldsymbol{\sigma}}$ must assume a functional behaviour of the parameter SE-continuations. Such properties have a pre- and post-condition flavour. One main challenge in the verification of continuation-based functions is to find an invariant strong enough to prove correctness. Let $S, e \models_{\boldsymbol{\phi}} f \leftrightarrow g$ be short for $\forall x \in S.(\!|f|\!)^{x\cdot e} \leftrightarrow (\!|g|\!)^{x\cdot e}$. In (4.87) we prove such a property for $\mathsf{split}_{\boldsymbol{\sigma}}$, but since the verification is not completed, we do not know if the invariant is strong enough. Probably not.

$$\begin{array}{r}S \neq \emptyset \wedge \mathsf{ishorn}\ p \wedge \mathsf{wf}\ se \wedge (\exists s.\mathsf{precise}\ s \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s) \wedge\\ (\forall s, se.\mathsf{wf}\, se \wedge S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se \wedge s \in \{\mathsf{P}, \mathsf{N}\} \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s \rightarrow S, e \models_{\boldsymbol{\phi}} f \leftrightarrow ct_{\not\mathsf{Z}}\ se) \wedge\\ (\forall se.\mathsf{wf}\, se \wedge S, e \models^{\boldsymbol{\rho}}_{[\boldsymbol{\sigma}]} se \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : \mathsf{Z} \rightarrow S, e \models_{\boldsymbol{\phi}} f \leftrightarrow ct_{\mathsf{Z}}\ se) \rightarrow\\ S, e \models_{\boldsymbol{\phi}} f \leftrightarrow \mathsf{split}_{\boldsymbol{\sigma}}\ p\ ct_{\mathsf{Z}}\ ct_{\not\mathsf{Z}}\ se\end{array} \quad (4.87)$$

Note that since $\mathsf{split}_{\boldsymbol{\sigma}}$ is always called on the head of some polynomial, which is constant, it is very easy to discharge the $\exists s.\mathsf{precise}\ s \wedge S, e \models^{\boldsymbol{\rho}}_{\boldsymbol{\sigma}} p : s$ assumption.

Challenges and more questions

In §4.5.2 we have proved all "mathematical" properties about polynomials we need, and we have presented means of transferring these properties to $\boldsymbol{\rho}$-polynomials. The remaining challenge in the verification of $\mathsf{qe}_{\mathbb{R}}$ is to find strong invariants, which accurately describe the functional behaviour of SE- and SM-continuations. The general problem of verifying continuation-based programs is an interesting challenge.

A further challenge and question is related to partial functions. So far, we have not proved that the main-loop functions terminate. Hence, from a HOL perspective, the equations in Figure 4.15 do not hold. Consequently, we can not generate ML code for the procedure, without *assuming* the equations. We can argue that this step is correct only in meta-theory (on paper, outside HOL). More generally, code-generation for partial functions is a very interesting topic, so far with no satisfying answers. Assume we have a HOL partial function f. Are we allowed to execute the definition of f on inputs x such that $x \in \mathsf{dom}(f)$? If yes, can we have a proof or proof-method in HOL?

4.5.4 Heuristics to reduce sign-assumptions

An undesirable property of the sign-splittings so far is that it splits over *all* signs blindly, where some of them can be safely omitted by syntactical checks. A square polynomial, e.g. y^2, will never take negative values, and hence assuming $y^2 < 0$ already leads to a contradiction. This should be detected at the splitting-stage. Note that this does not only influence the run-time of the qep., but also the size of the qe. result.

We subsequently consider a few simple syntactical tests according to which some splittings can be safely avoided and prove this new splitting approach correct.

Assume we have a heuristic function h that given a SE se, a polynomial p and a sign s returns *True only if* $se[p \leftarrow s]$ is inconsistent, then we can parametrise sign assertions by h. Note that the constant function $h_0 = \lambda se, q, s.False$ is a correct yet useless heuristic ($se[p \leftarrow s]_{h_0} = se[p \leftarrow s]$). We prove the generic property (4.88).

$$\begin{aligned} se[p \leftarrow s]_h \quad = \quad & \textbf{if } p = \mathbf{0}_{\boldsymbol{\rho}} \textbf{ then if } s = \mathsf{Z} \textbf{ then } \lfloor se \rfloor \textbf{else} \bot \\ & \textbf{else let } (p', b) = \mathsf{monic}\ p\ ;\ s' = \mathsf{swap}_{\boldsymbol{\sigma}}\ b\ s\ ; \\ & \qquad t = (\textbf{case } \overrightarrow{se}\ p' \textbf{ of } \lfloor t' \rfloor \Rightarrow t' \mid \bot \Rightarrow s') \\ & \quad \textbf{in if } s' = t \wedge \neg h\ se\ p'\ s' \textbf{ then } \lfloor (p', s') \cdot (\mathsf{del}\ p'\ se) \rfloor \textbf{ else } \bot \end{aligned}$$

$$\begin{aligned} & S \neq \emptyset \wedge \mathsf{precise}\ s \wedge \mathsf{ishorn}\ p \wedge \mathsf{wf}\ se \qquad (4.88) \\ \wedge (\forall S, se, p, s, e.\mathsf{wf}\ se \wedge S \neq \emptyset \wedge \mathsf{ishorn}\ p \wedge h\ se\ p\ s \rightarrow \neg(S \models_e se \wedge S \models_e p : s) & \\ \rightarrow (\textbf{case } se[p \leftarrow s]_h \textbf{ of } \bot \Rightarrow False \mid \lfloor se_0 \rfloor \Rightarrow S \models_e se_0) \leftrightarrow S \models_e se \wedge S \models_e p : s & \end{aligned}$$

In the following we develop an instance of h. Let us first consider simple criteria for $p :: \boldsymbol{\rho}$ to be zero ($p{=}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$), non-positive ($p{\leq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$) or non-negative ($p{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$) in a SE se. If $p = \mathbf{0}_{\boldsymbol{\rho}}$ or $\mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \lfloor \mathsf{Z} \rfloor$ then clearly p is zero, i.e.

$$p{=}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} = (p = \mathbf{0}_{\boldsymbol{\rho}} \vee \mathsf{find}_{\boldsymbol{\sigma}}\ se\ p = \lfloor \mathsf{Z} \rfloor) \qquad (4.89)$$

Note that we can test non-positivity and non-negativity of a constant polynomial $\boldsymbol{r}_{\boldsymbol{\rho}}$ directly. Assume p has the form $c + \boldsymbol{v}_n * d + \boldsymbol{v}_m * q$, and consider the two cases $n = m$ and $n \neq m$. If $n = m$, then a sufficient non-negativity condition for p is that $c{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge q{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$ and either $d{=}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$ or $d{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge \boldsymbol{v}_n{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$ or $d{\leq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge \boldsymbol{v}_n{\leq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$. Now assume $n \neq m$. If there is no assumption on the sign of $\boldsymbol{v}_n$ in se, then a sufficient condition is $c{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge d + \boldsymbol{v}_m * q{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$. Assume $\boldsymbol{v}_n$ is assigned a sign s in se. If $s = \mathsf{Z}$ then we require $c{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$, if $s = \mathsf{P}$ then $c{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge d + \boldsymbol{v}_m * q{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$ is sufficient and if $s = \mathsf{N}$ then $c{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \wedge d + \boldsymbol{v}_m * q{\leq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}}$ is sufficient. We define the other cases analogously. Non-positivity proceeds similarly. Note that the definition of non-positivity and that of non-negativity are mutually recursive. We omit the definitions since these are lengthy and technical, while their idea is already captured by the previous sketch. Using these functions we define a simple heuristic h and prove it correct:

$$\mathsf{h}\ se\ p\ s = \textbf{case } s \textbf{ of } \mathsf{Z} \Rightarrow False \mid \mathsf{P} \Rightarrow p{\leq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \mid \mathsf{N} \Rightarrow p{\geq}^?_{se}\mathbf{0}_{\boldsymbol{\rho}} \mid \_ \Rightarrow False \qquad (4.90)$$

$$\forall S, se, p, s, e.\mathsf{wf}\ se \wedge S \neq \emptyset \wedge \mathsf{ishorn}\ p \wedge \mathsf{h}\ se\ p\ s \rightarrow \neg(S \models_e se \wedge S \models_e p : s) \qquad (4.91)$$

Ultimately (4.81) holds, when $se[p \leftarrow s]$ is replaced by $se[p \leftarrow s]_{\mathsf{h}}$.

4.5.5 Optimisations using linear and quadratic equations

In the following we show how to exploit the following rule, which leads to *immediate* elimination of the quantifier.

$$(\exists x.x = t \wedge P(x)) \leftrightarrow P(t).$$

Of course in practice we expect that equality $x = t$ should be *recognised* within all conjuncts under $\exists$, i.e. it does not have to appear immediately after $\exists$ and it could also be $t = x$,

or $5 \cdot x + y - z = 3 \cdot z$, for this is equivalent to $x = \frac{4 \cdot z - y}{5}$. Recognising equations containing the bound variable *linearly* is simple. Note that it is not trivial to substitute e.g. $\frac{z^2+y}{y^2+4 \cdot z}$ for x only using the language of rings. This is subsumed by the techniques we consider in this section.

In the following we show how to use a *quadratic equation* to optimise qe., i.e. we treat formulae of the form $\exists x.a \cdot x^2 + b \cdot x + c = 0 \wedge P(x)$. This case is practically relevant, see [Wei97b] upon which we base the following formalisation.

The solution of the quadratic equation over the reals is expressed in (4.92) or equivalently in (4.93) in terms of the discriminant $\Delta = b^2 - 4 \cdot a \cdot c$ and $\delta = \sqrt{\Delta}$.

$$\begin{aligned} a \cdot x^2 + b \cdot x + c = 0 \leftrightarrow a = 0 \wedge (b = c = 0 \vee b \neq 0 \wedge x = \frac{-c}{b}) \vee \\ a \neq 0 \wedge \Delta \geq 0 \wedge (x = \frac{-b - \delta}{2 \cdot a} \vee x = \frac{-b + \delta}{2 \cdot a}) \end{aligned} \tag{4.92}$$

$$\begin{aligned} (\exists x.a \cdot x^2 + b \cdot x + c = 0 \wedge P\ x) \leftrightarrow a = 0 \wedge (b = c = 0 \wedge (\exists x.P\ x) \vee b \neq 0 \wedge P(\frac{-c}{b})) \vee \\ a \neq 0 \wedge \Delta \geq 0 \wedge (P(\frac{-b - \delta}{2 \cdot a}) \vee P(\frac{-b + \delta}{2 \cdot a})). \end{aligned} \tag{4.93}$$

Assume that $P(\frac{-c}{b})$ and $P(\frac{-b \pm \sqrt{\Delta}}{2 \cdot a})$ have qf. equivalent formulae in our language for $\mathfrak{R}$, then (4.93) ultimately yields qe. for $\exists x.a \cdot x^2 + b \cdot x + c = 0 \wedge P(x)$ *under the condition that* $\neg(a = b = c = 0)$ and that P is qf. Recall that the coefficients a, b and c are generally multivariate polynomials in the remaining variables. Weispfenning gave in [Wei97b] an algorithm to construct qf. formulae equivalent to $P(\frac{-c}{b})$ and $P(\frac{-b \pm \sqrt{\Delta}}{2 \cdot a})$, which we formalise in the following.

Fractions and their virtual substitution

In §4.3.1 we have already developed a virtual substitution of fractions, but there we considered only linear polynomials and the problem was much simpler. Here we need to substitute fractions into polynomial expressions (this yields a fraction) and then transform the atoms accordingly. As it will soon become clear, we only need to manipulate fractions with the same denominator, taken to different powers though. Let $\frac{p}{q^n}$ denote the fraction of the $\boldsymbol{\rho}$-polynomial p by the $\boldsymbol{\rho}$-polynomial q to the power of n. Formally these are just triples interpreted by $(\!|\frac{p}{q^n}|\!)^e_f$ (cf. (4.94)) and which we call normal and compatible if they satisfy isnorm^e and $\equiv_/$ in (4.95) and (4.96), respectively.

$$(\!|\frac{p}{q^n}|\!)^e_f = \frac{(\!|p|\!)^e_{\boldsymbol{\rho}}}{((\!|q|\!)^e_{\boldsymbol{\rho}})^n} \tag{4.94}$$

$$\mathsf{isnorm}^e\ \frac{p}{q^n} = \mathsf{unbound}_{\boldsymbol{\rho}}\ p \wedge \mathsf{unbound}_{\boldsymbol{\rho}}\ q \wedge \mathsf{ishorn}\ p \wedge \mathsf{ishorn}\ q \wedge ((\!|q|\!)^e_{\boldsymbol{\rho}})^n \neq 0 \tag{4.95}$$

$$\frac{a}{b^n} \equiv_/ \frac{c}{d^m} \leftrightarrow b = d \tag{4.96}$$

We define addition $(\frac{a}{b^n} +_f \frac{c}{d^m})$ and multiplication $(\frac{a}{b^n} *_f \frac{c}{d^m})$ of two normal and compatible fractions $\frac{a}{b^n}$ and $\frac{c}{d^m}$ in (4.97) and (4.98):

$$\frac{a}{b^n} +_f \frac{c}{d^m} = \mathbf{let}\ k = \max\ m\ n; l = \min\ m\ n\ \mathbf{in}\ \frac{b^{\downarrow \boldsymbol{m-l}} \circledast a \oplus b^{\downarrow \boldsymbol{n-l}} \circledast c}{b^k} \tag{4.97}$$

$$\frac{a}{b^n} *_f \frac{c}{d^m} = \frac{a \circledast c}{b^{n+m}}. \tag{4.98}$$

Now we can define the substitution $p[\frac{a}{b^n}]_{\boldsymbol{\rho}}$ of a normal fraction $\frac{a}{b^n}$ for the bound variable $\boldsymbol{v}_0$ in a $\boldsymbol{\rho}$-polynomial p (just recursively apply addition and multiplication of fractions). Note

that the result is a fraction. The following key properties (4.99) and (4.100) state that these operations preserve the semantics, normal form and compatibility:

$$\begin{aligned}\mathsf{isnorm}^e\ \frac{a}{b^n} \wedge \mathsf{isnorm}^e\ \frac{c}{d^m} \wedge \frac{a}{b^n} \equiv_/ \frac{c}{d^m} \rightarrow \\ \mathsf{isnorm}^e\ (\frac{a}{b^n}\ \beta\ \frac{c}{d^m}) \wedge (\!|\frac{a}{b^n}\ \beta\ \frac{c}{d^m}|\!)^e_f = (\!|\frac{a}{b^n}|\!)^e_f \circ (\!|\frac{c}{d^m}|\!)^e_f \wedge \frac{a}{b^n}\ \beta\ \frac{c}{d^m} \equiv_/ \frac{a}{b^n} \\ \text{for } (\beta, \circ) \in \{(+_f, +), (*_f, \cdot)\}\end{aligned} \tag{4.99}$$

$$\mathsf{isnorm}^{x\cdot e}\ \frac{a}{b^n} \wedge \mathsf{ishorn}\ p \rightarrow \mathsf{isnorm}^{x\cdot e}\ p[\frac{a}{b^n}]_{\boldsymbol{\rho}} \wedge (\!|p[\frac{a}{b^n}]_{\boldsymbol{\rho}}|\!)^{x\cdot e}_f = (\!|p|\!)^{((\!|\frac{a}{b^n}|\!)^{x\cdot e}_f)\cdot e}_{\boldsymbol{\rho}} \tag{4.100}$$

Now the substitution $f[\frac{a}{b^n}]_{\phi}$ of a normal fraction $\frac{a}{b^n}$ into a formula f is simple: for an atom $q \bowtie \mathbf{0}_{\boldsymbol{\rho}}$, where $\bowtie \in \{<, \leq, =\}$, compute the fraction $\frac{p}{b^k} = q[\frac{a}{b^n}]_{\boldsymbol{\rho}}$ and return

$$\begin{cases} p = \mathbf{0}_{\boldsymbol{\rho}} & \text{for } \bowtie = = \\ p \geq \mathbf{0}_{\boldsymbol{\rho}} \wedge b^{\downarrow \boldsymbol{\delta_k}} \leq \mathbf{0}_{\boldsymbol{\rho}} \vee p \leq \mathbf{0}_{\boldsymbol{\rho}} \wedge b^{\downarrow \boldsymbol{\delta_k}} \geq \mathbf{0}_{\boldsymbol{\rho}} & \text{for } \bowtie = \leq \\ p > \mathbf{0}_{\boldsymbol{\rho}} \wedge b^{\downarrow \boldsymbol{\delta_k}} < \mathbf{0}_{\boldsymbol{\rho}} \vee p < \mathbf{0}_{\boldsymbol{\rho}} \wedge b^{\downarrow \boldsymbol{\delta_k}} > \mathbf{0}_{\boldsymbol{\rho}} & \text{for } \bowtie = <, \end{cases}$$

where δ_k formalises Kronecker δ for $\mathsf{odd}\ k$, i.e. $\delta_k = 0$ if k is even and 1 otherwise, for any $k :: \mathbb{N}$. We finish the substitution of fractions by proving the main property in (4.101).

$$\begin{aligned}\mathsf{qfree}\ p \wedge \mathsf{isnorm}^{x\cdot e}\ \frac{a}{b^n} \wedge \mathsf{all}_{\boldsymbol{\rho}}\ \mathsf{ishorn}\ p \rightarrow \\ (\!|p[\frac{a}{b^n}]_{\phi}|\!)^{x\cdot e} \leftrightarrow (\!|p|\!)^{((\!|\frac{a}{b^n}|\!)^{x\cdot e}_f)\cdot e}_{\boldsymbol{\rho}} \wedge \mathsf{unbound}_{\phi}\ (p[\frac{a}{b^n}]_{\phi}).\end{aligned} \tag{4.101}$$

Surds and their virtual substitution

Let a surd expression be $d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})$ for a natural number k and $\boldsymbol{\rho}$-polynomials a, b, c and d. Formally this is just a 5-tuple interpreted in an environment e by $(\!|d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\surd}$ (cf. (4.102)) and which we call normal and compatible with $l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})$ if it respectively satisfies isnorm^e in (4.103) and $d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) \equiv_{\surd} l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})$ in (4.104).

$$(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\surd} = \frac{(\!|a|\!)^e_{\boldsymbol{\rho}} + (\!|b|\!)^e_{\boldsymbol{\rho}} \cdot \sqrt{(\!|c|\!)^e_{\boldsymbol{\rho}}}}{((\!|d|\!)^e_{\boldsymbol{\rho}})^k} \tag{4.102}$$

$$\mathsf{isnorm}^e\ d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}}) = (\!|d|\!)^e_{\boldsymbol{\rho}} \neq 0 \wedge (\!|c|\!)^e_{\boldsymbol{\rho}} \geq 0$$

$$\wedge \forall q \in \{a, b, c, d\}.\mathsf{ishorn}\ q \wedge \mathsf{unbound}_{\boldsymbol{\rho}}\ q \tag{4.103}$$

$$d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) \equiv_{\surd} l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}}) = c = h \wedge d = l \tag{4.104}$$

As before, we define the addition $+_{\surd}$ and multiplication $*_{\surd}$ of two normalised and compatible surd expressions in (4.105) and (4.106), respectively.

$$\begin{aligned}d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) +_{\surd} l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}}) = \\ \mathbf{let}\ k = \max\ k_1\ k_2\ ;\ d_1 = d^{\downarrow(\boldsymbol{k-k_1})}\ ;\ d_2 = d^{\downarrow(\boldsymbol{k-k_2})} \\ \mathbf{in}\ d^{-\boldsymbol{k}}((d_1 \circledast a \oplus d_2 \circledast f) + (d_2 \circledast b \oplus g \circledast d_1)c^{\frac{1}{\boldsymbol{2}}})\end{aligned} \tag{4.105}$$

$$\begin{aligned}d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) *_{\surd} l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}}) = \\ d^{-(\boldsymbol{k_1+k_2})}((a \circledast f \oplus b \circledast g \circledast c) + (a \circledast g \oplus f \circledast b)c^{\frac{1}{\boldsymbol{2}}})\end{aligned} \tag{4.106}$$

It is an easy exercise to prove the previous operations correct, cf. (4.107) and to define the substitution $p[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\rho}}$ of a normalised surd $d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})$ into a $\boldsymbol{\rho}$-polynomial p and prove it correct, cf. (4.108).

$$\begin{aligned}\mathsf{isnorm}^e\ d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) \wedge \mathsf{isnorm}^e\ l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}}) \wedge d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) \equiv_{\bigvee} l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})\\ \rightarrow (\!|d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}})\,\beta\, l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} = (\!|d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} \circ (\!|l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee}\\ \wedge \mathsf{isnorm}^e\ (d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}})\,\beta\, l^{-\boldsymbol{k_2}}(f + gh^{\frac{1}{\boldsymbol{2}}})) \quad \text{for } (\beta,\circ) \in \{(+_{\bigvee},+),(*_{\bigvee},\cdot)\}\end{aligned} \tag{4.107}$$

$$\begin{aligned}\mathsf{isnorm}^{x\cdot e}\ d^{-\boldsymbol{k_1}}(a + bc^{\frac{1}{\boldsymbol{2}}}) \wedge \mathsf{ishorn}\ p \rightarrow\\ \mathsf{isnorm}^e\ p[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\rho}} \wedge (\!|p[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\rho}}|\!)^{x\cdot vs}_{\bigvee} = (\!|p|\!)^{((\!|d^{-\boldsymbol{k}}(a+bc^{\frac{1}{\boldsymbol{2}}})|\!)^{x\cdot e}_{\bigvee})\cdot e}_{\boldsymbol{\rho}}\end{aligned} \tag{4.108}$$

Now we are ready for the substitution $f[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\phi}}$ of a normalised surd expression $d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})$ into a formula f. Similarly to the previous case of fractions, it is sufficient to construct formulae semantically equivalent to $(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee}\ \beta\ 0$, for $\beta \in \{=,<,\leq\}$. First consider the simple case where $(\!|b|\!)^e_{\boldsymbol{\rho}} = 0$. Then the following holds:

$$(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} = 0 \leftrightarrow (\!|a|\!)^e_{\boldsymbol{\rho}} = 0 \tag{4.109}$$

$$(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} \leq 0 \leftrightarrow (\!|a \circledast d^{\downarrow\boldsymbol{\delta_k}}|\!)^e_{\boldsymbol{\rho}} \leq 0 \tag{4.110}$$

$$(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} < 0 \leftrightarrow (\!|a \circledast d^{\downarrow\boldsymbol{\delta_k}}|\!)^e_{\boldsymbol{\rho}} < 0 \tag{4.111}$$

In the general case, the following holds:

$$(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} = 0 \leftrightarrow (\!|a \circledast b \leq \mathbf{0_\rho} \wedge a^{\downarrow\boldsymbol{2}} \ominus b^{\downarrow\boldsymbol{2}} \circledast c = \mathbf{0_\rho}|\!)^e \tag{4.112}$$

$$\begin{aligned}(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} \leq 0 \leftrightarrow (\!|a \circledast d^{\downarrow\boldsymbol{\delta_k}} \leq \mathbf{0_\rho} \wedge a^{\downarrow\boldsymbol{2}} \ominus b^{\downarrow\boldsymbol{2}} \circledast c \geq \mathbf{0_\rho}|\!)^e\\ \vee (\!|b \circledast d^{\downarrow\boldsymbol{\delta_k}} \leq \mathbf{0_\rho} \wedge a^{\downarrow\boldsymbol{2}} \ominus b^{\downarrow\boldsymbol{2}} \circledast c \leq \mathbf{0_\rho}|\!)^e\end{aligned} \tag{4.113}$$

$$\begin{aligned}(\!|d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})|\!)^e_{\bigvee} < 0 \leftrightarrow (\!|a \circledast d^{\downarrow\boldsymbol{\delta_k}} < \mathbf{0_\rho} \wedge a^{\downarrow\boldsymbol{2}} \ominus b^{\downarrow\boldsymbol{2}} \circledast c > \mathbf{0_\rho}|\!)^e\\ \vee (\!|b \circledast d^{\downarrow\boldsymbol{\delta_k}} \leq \mathbf{0_\rho} \wedge (a \circledast d^{\downarrow\boldsymbol{\delta_k}} < \mathbf{0_\rho} \vee a^{\downarrow\boldsymbol{2}} \ominus b^{\downarrow\boldsymbol{2}} \circledast c < \mathbf{0_\rho})|\!)^e\end{aligned} \tag{4.114}$$

The proofs proceed by case distinctions on the signs. We encourage the reader to do the proof of (4.114). Finally we prove the desired property for substitution of surd expressions into formulae:

$$\begin{aligned}\mathsf{qfree}\ p \wedge \mathsf{isnorm}^{x\cdot e}\ s \wedge \mathsf{all}_{\boldsymbol{\rho}}\ \mathsf{ishorn}\ p \rightarrow\\ \mathsf{unbound}_{\boldsymbol{\phi}}\ p[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\phi}} \wedge (\!|p[d^{-\boldsymbol{k}}(a + bc^{\frac{1}{\boldsymbol{2}}})]_{\boldsymbol{\phi}}|\!)^{x\cdot e} \leftrightarrow (\!|p|\!)^{((\!|d^{-\boldsymbol{k}}(a+bc^{\frac{1}{\boldsymbol{2}}})|\!)^{x\cdot e}_{\bigvee})\cdot e}.\end{aligned} \tag{4.115}$$

The overall method

The overall method quad (see Figure 4.18) lifts a qep. qe to an optimising one using one quadratic or linear equation. Given a $\boldsymbol{\phi}$-formula p, it first decomposes it into its conjuncts (using $\mathsf{split}_\wedge$), and then looks for a linear or quadratic equation (using isquad). If no such equation occurs then we just apply qe otherwise we apply (4.93) using our virtual substitution machinery for fractions and surds.

The main property of quad is expressed in (4.116). Note that when using a linear equation there is no loss of efficiency or conciseness of the formula, since the $a' = \mathbf{0_\rho}$ in Figure 4.18 immediately evaluates to $\boldsymbol{F}$ and hence the whole conjunction is discarded. Recall that we always use the optimised constructors. The whole formalisation of this optimisation took 1600 lines of Isabelle proofs.

$$
\begin{array}{lcl}
\mathsf{isquad}\ f & = & \mathbf{case}\ f\ \mathbf{of}\ p = \mathbf{0_\rho} \Rightarrow \mathsf{deg}_{\boldsymbol{\rho}}\ 0\ p \in \{\!\{1,2\}\!\} \quad \mid \Rightarrow \mathit{False} \\
\mathsf{quadcoeffs}\ p & = & \mathbf{case}\ p\ \mathbf{of}\ c + \boldsymbol{v}_0 * (b + \boldsymbol{v}_0 * a) = \mathbf{0_\rho} \Rightarrow (a,b,c) \\
 & & \quad \mid\ c + \boldsymbol{v}_0 * b = \mathbf{0_\rho} \Rightarrow (\mathbf{0_\rho}, b, c) \\
\mathsf{quadqe}\ qe\ (a,b,c)\ p & = & \mathbf{let}\ \Delta = b^{\downarrow \mathbf{2}} \ominus \mathbf{4_\rho} \circledast a \circledast c; [a',b',c',\delta] = \mathsf{map}\ \mathsf{decr}_{\boldsymbol{\rho}}\ [a,b,c,d] \\
 & & \mathbf{in}\ a' = \mathbf{0_\rho} \wedge (b' = c' = \mathbf{0_\rho} \wedge (qe\ p) \vee b' \neq \mathbf{0_\rho} \wedge \mathsf{decr}_{\boldsymbol{\phi}}(p[\frac{\ominus c}{b^{\mathbf{1}}}]_{\boldsymbol{\phi}})) \\
 & & \quad \vee a' \neq \mathbf{0_\rho} \wedge \delta \geq \mathbf{0_\rho} \wedge \mathsf{decr}_{\boldsymbol{\phi}}(p[(\mathbf{2_\rho} \circledast a)^{-\mathbf{1}}(\ominus b + \mathbf{1_\rho}\Delta^{\frac{\mathbf{1}}{\mathbf{2}}})]_{\boldsymbol{\phi}} \\
 & & \quad \vee p[(\mathbf{2_\rho} \circledast a)^{-\mathbf{1}}(\ominus b + -\mathbf{1_\rho}\Delta^{\frac{\mathbf{1}}{\mathbf{2}}})]_{\boldsymbol{\phi}}) \\
\mathsf{quad}\ qe\ p & = & \mathbf{let}\ cjs = \mathsf{remdups}(\mathsf{split}_\wedge\ p) \\
 & & \mathbf{in\ case}\ \mathsf{get\_index}\ \mathsf{isquadeq}\ cjs\ \mathbf{of}\ \bot \Rightarrow\ qe\ p \\
 & & \mid\ \lfloor i \rfloor \Rightarrow\ \mathbf{let}\ eq = cjs!i\ ;\ cjs' = \mathsf{remove}\ eq\ cjs \\
 & & \quad \mathbf{in}\ \mathsf{quadqe}\ qe\ (\mathsf{quadcoeffs}\ eq)(\mathsf{list}_\wedge\ cjs')
\end{array}
$$

Figure 4.18: Optimisation using quadratic equations

$$
\begin{array}{r}
(\forall e, p.\mathsf{qfree}\ p \rightarrow \mathsf{qfree}\ (qe\ p) \wedge (\!|qe\ p|\!)^e \leftrightarrow (\!|\exists\, p|\!)^e) \wedge \mathsf{qfree}\ p \wedge \mathsf{all}_{\boldsymbol{\rho}}\ \mathsf{ishorn}\ p \\
\rightarrow \mathsf{qfree}(\mathsf{quad}\ qe\ p) \wedge (\!|\mathsf{quad}\ qe\ p|\!)^e \leftrightarrow (\!|\exists\, p|\!)^e \qquad (4.116)
\end{array}
$$

How ad-hoc are Weispfenning's considerations for special cases?

The key idea of the previous development is the following. Given a constraining equation $p = 0$, we express *all* solutions to $p = 0$ using our language and radicals and perform a modified substitution for any solution. Since the general polynomial equation of degree n is not solvable for $n > 4$ by the Abel-Ruffini theorem (cf. [Art98]) the described method cannot be generalised to arbitrary equations. Weispfenning considered the cubic and quartic cases [Wei94b]. It would be interesting to pursue this approach for $n > 4$ where the Galois group of p is solvable.

4.6 Related work

A brief history of qe.:

Quantifier elimination enjoys a very exciting history and can be traced back at least to the work of Descartes, Boudin, Fourier, Sturm and Sylvester on counting the number of real roots of polynomials, cf. [DSW98b]. In modern logic the problem of quantifier elimination seems to originate in Boole's and Schröders's work as the elimination of middle terms from (equational) hypotheses in the calculus of classes. First in the pioneer work of Löwenheim [Löw15], where he proves a weaker form of the Löwenheim-Skolem theorem, we find initial attempts to qe. Skolem [Sko19, Sko70a] soon recognised the importance of [Löw15] and extended its results using an alternative proof. Skolem's paper [Sko19] is highly interesting, since it presents a formal treatment of qe., lifting properties and actually proves the first qe. result: "the first-order theory of the calculus of classes with numerical predicates admits qe.". See also [Sko70a] for his generalisation of Löwenheim's [Löw15] theorem, where he also introduces what we know now as Morleysation of a theory. Skolemisation can be seen as an abstract qep. For the cases where the Skolem functions can only take some values expressible in the underlying language, the resulting device is qe. by terms, used later by Cooper, Ferrante and Rackoff and generalised by Weispfenning.

Tarski is a major figure in qe.'s history. He soon realised the importance of [Löw15, Sko19, Sko70a] and held many lectures on qe. After Langford's qep. [Lan27, Sko28] for dense linear orders without endpoints, he gave an "exercise" to his seminar student Presburger. The result of this exercise is now known as the completeness and qe. of Presburger Arithmetic [Pre29]. Presburger did not consider ordering in [Pre29] but published later an addendum that his proofs can be easily modified to deal with ordering. Tarski did not recognise this

result worth a PhD , cf. [Smo91]. Apparently [Wei06] (Weispfenning was told this story but was not absolutely sure himself), poor Presburger did not solve the assigned problem: a qep. for $\mathbb{Z}$ with *both* addition and multiplication. One year after Presburger's paper, Skolem [Sko30] independently arrived at a qep. for linear arithmetic over $\mathbb{Z}$ quite differently: he uses the $\lfloor\rfloor$-function and multiplication by rational constants. See [Smo91] for a comparison of Preburger's and Skolem's qe. procedures. It is also interesting to compare some small similarities of [Sko30] to Weispfenning's qep. for mixed real-integer arithmetic in [Wei99]. In the same paper Skolem proved Peano arithmetic without addition, now known as Skolem arithmetic, decidable.

Tarski proved several qe. results, e.g. [TM49, Tar51], but the most famous one is RCF [Tar51]. He had a qep. in round 1930, but this result remained unpublished until 1948. In [Tar31, KT31] he hints to his qe. result and uses it to study definable sets in RCF. Tarski's (and his many students) extensive work on (eastern) model theory made qe., as a method, an extremely powerful tool to study particular theories. See [Wei83] for a survey on qe. in algebraic structures and its connection to model theory. See e.g. [Rob49] for impressive negative decidability results obtained by studying definability. The interest of other communities in qe. had first to await computers.

Complexity results and algorithms for qe.

Round 1970, there was an intense revived interest in qe. and decision procedures for algebraic theories. This is mainly due to complexity studies and the interest of automated reasoning in the qe. and decision problem. This is an important shift of interest: it is not sufficient to *know* about a qe. result, but the qe. algorithms themselves are important. Before 1970 most qe. algorithms we non-elementary in complexity, e.g. [Tar51] and Cohen's simpler procedure [Coh69] for RCF. See also [KK67] for qe. procedures for RCF, ACF, DLO and Presburger arithmetic.

Linear arithmetic: The first theorem prover [Dav57] is an implementation of Presburger's algorithm. See also [KK67, HB68] for alternative presentations of Presburger's algorithm. Unsatisfied with the bad complexity behaviour of Presburger's algorithm in automated program verification, Cooper [Coo72] published a new substantially more efficient qep. for $\mathcal{Z}_+$. Apparently, this paper plays a tun-point in the history of qe.. Oppen [Opp73] proved that Cooper's algorithm is triply exponential in time. In 1974 Fisher and Rabin [FR74] published a fundamental and striking result about lower complexity bounds for linear arithmetic: single and doubly exponential in space for $\mathbb{R}$ and $\mathbb{Z}$ respectively. Cooper's ideas of avoiding DNF and introducing formulae to simulate the behaviour of the input near infinity is crucial. This inspired Ferrante and Rackoff for a qe. for $\mathcal{R}_+$ [FR75]. See also [FR79] for a general framework for decision procedures based on Ehrenfeucht-games together with generic complexity results. Weispfenning [Wei88] generalised the underlying device by introducing virtual substitution of impure terms (like $+\infty$ and $-\infty$). He also gave more efficient qe. procedures for $\mathcal{Z}_+$ and $\mathcal{R}_+$ in [Wei88, Wei90] and also efficient qe. procedures that deal with parameters [LW93, Wei90, Wei97a]. Berman [Ber77] proved that even the combination of space and time, as complexity measures, is not enough to capture the exact complexity behaviour of linear arithmetic. They must be combined with alternation, see also [Ber80]. Reddy and Loveland [RL78] gave the first complexity-optimal procedure for $\mathcal{Z}_+$ in Berman's sense. Complexity-optimal decision and qe. procedures for linear arithmetic over ordered fields and $\mathbb{Z}$ can be found in [Wei88] and [Wei90] together with the first complexity studies of the qe. problem, which is one exponent worse than the decision problem. See also [Wei97a] for a (weak) qep. for Presburger arithmetic with non-linear parameters (also called uniform Presburger arithmetic), using bounded quantifiers. Especially [Wei88] is very interesting for the first introduction of the virtual substitution device and also for several generic complexity theorems for the qe. and decision problems for linear arithmetic over ordered fields. Weispfenning, Dolzmann and Sturm used this device to develop many impressively efficient qep. in REDLOG [DS97] for parametric linear arithmetic over ordered fields [LW93],

the quadratic [Wei94a, Wei97b] and cubic [Wei94b] case of real algebra (see [DSW98a] for a very interesting adaptation to theorem proving in geometry) and the theory of mixed real-integer linear arithmetic in [Wei99]. See [Wei00, AW00] for even harder theories of linear real arithmetics with transcendental (axiomatised in a manner to subsume exponential and hyperbolic functions and the arcus tangus) or with trigonometric functions. These qe. procedures in contrast to previous results do not rely on Shanuel's conjecture but only on Lindemann's theorem from transcendence theory. One major strength of Weispfenning's approach to qe. is that it computes *answers* for purely existential problems. This is very useful for a certificate-based integration in theorem provers. For efficient automata-based alternatives for linear arithmetic see [WB95, BJW05] and [Kla04] for an optimal complexity upper bound. This approach represents a formula ϕ by an automata whose accepted language is exactly all the values to the free variables satisfying ϕ. All logical operations reduce to operations on automata. Testing satisfiability of the input formula reduces to an emptiness test of the resulting automata. This approach is potentially useful for a certificate based integration for universal or existential formulae: if the resulting automaton is not empty then produce an element, this a witness or a counter example, respectively. See also [GBD02] for a SAT-model-checking approach to $\mathcal{Z}_+$.

Real closed fields Tarski's qep. for RCF was complicated and non-elementary. Cohen [Coh69] gave a simpler qep., which has been further simplified in [Hör83]. This procedure also has non-elementary complexity. The first doubly exponential qep. for RCF was CAD, apparently found by Collins round 1973 and published in [Col75]. Independently, Monk [Mon74] found a decision procedure for RCF, which was one exponent worse than CAD. Solovay changed Monk's procedure in 1975 to meet CAD's complexity. This result is apparently not published but mentioned in [Col75], as a letter communication, and by Wüthrich [Wüt74] where he obtains a qep. for RCF based on the decision procedure. Collins's CAD was the first qep. for RCF to be implemented on a computer, and is until today one of the most efficient algorithms in practice. See [Wei98] for a competitive approach, based on comprehensive Gröbner bases. In [BOKR84], the authors show that deciding RCF is complete for EXPSPACE. Their result also applies to ACF. An other surprising result is that qe. for RCF is doubly exponential, see [DH88, Wei88]. This means that qe. for RCF is, computationally, not harder than qe. over ordered fields. There are many subtly different upper bounds on the complexity of deciding RCF, see e.g. [Gri88, GV88] or [Ren92a, Ren92b, Ren92c] for an overview.

Algebraically closed fields The history of ACF is not very clear. The qe. result is often attributed to Tarski, but I was not able to find a convincing reference. In [Tar51], Tarski points, as possible future work, that his results should easily extend to the complex numbers where ordering holds only if the numbers are real. Note that a decision procedure for RCF automatically yields a decision procedure for the complex numbers. The earliest algorithmic description I was able to find is [KK67], which is non-elementary. The first doubly exponential qep. for ACF appears in [HW75, Wüt77]. See also [Hei83, Hei85] for more detailed presentation. For a fixed number of quantifier alternations the procedure in [CG84] is one exponent better.

Q.e. in theorem proving:

In the context of theorem proving, qe. have unfortunately not received enough attention, maybe due to the very bad complexity results. Most theorem provers implement Fourier's method for universal problems over $\mathbb{R}$. Note that it also works for $\mathbb{Z}$ since the problems are universal. The first qe. related work seems to be Harrison's PhD [Har96], where he integrated a proof procedure for RCF based on [KK67]. He also first gave a qep. for ACF in HOL in [Har01]. Théry [Thé] formalised Presburger's original algorithm in Coq in 2002. The first implementation and exposition of a complete method for Presburger arithmetic (Cooper's method and Omega test) in an LCF-like theorem prover (HOL) is due to Norrish

[Nor03]. Omega test [Pug91] is a variation on Fourrier's method to be complete on the integers. It has bad theoretical properties (it works on DNF) but performs very good on several practical problems. We also implemented Cooper's method in [Cha03] as a derived rule in Isabelle/HOL. In Coq an incomplete version of Omega test is due to [Cré04], but it only deals with universal formulae, and even there, it is incomplete. See also [BGD03] for an integration of a proof producing extension of Omega test for quantifier free mixed real and integer constraints in CVC. In [CN05] we gave the first full formalisation (reflection) of Cooper's algorithm, in [Cha06a] the first integration of Ferrante and Rackoff's procedure both as a derived rule and using reflection, and in [Cha06b] the first formalisation of a qep. for mixed real-integer linear arithmetic. An integration of Cohen's (or Hörmander's) procedure for real closed fields in HOL is presented in [MH05]. A far more ambitious work is Mahboubi's PhD [Mah06a] aiming to formally verify CAD in Coq. Unfortunately the formalisation is not complete but see [Mah06b, Mah06a] for impressive progress in gcd computation and the sub-resultants algorithm. In [DM06] the authors present a "qep." for "ACF" using Maple in a skeptical manner to compute gcds. The method there although original, can hardly be described as a qep., since it only deals with univariate polynomials, where the coefficients lie in a *decidable field*. The theory is hence different from ACF. We are not aware of any implementation of a qep. for ACF substantially more efficient than [KK67], e.g. [HW75, Wüt77, CG84] needs Bezout's theorem on intersections of algebraic curves, a far less trivial theorem to prove formally than the fundamental theorem of algebra.

Chapter 5

Conclusion

The goal of this thesis was to study the following question:

> *How should we integrate proof-procedures into an LCF-like system both in general and for particular relevant theories of arithmetic?*

We have studied the *in general* part in Chapter 2 and the *in particular* through the thesis, but especially in Chapter 3 and in Chapter 4. Figure 5.1 gives an overview of all proof methods, we have presented in the thesis, and the used integration paradigm. The latter part of the question is more technical but also easier to answer than the former. Like for most hard Software engineering problems, it is difficult to answer the "in general" part of our question categorically. I personally do not believe that it is possible. For particular cases, however, we can most of the time argue that one paradigm is better than another by measuring efficiency, implementation time etc. We give in the following a *heuristic* how to integrate a proof procedure in an LCF-like theorem prover. This is is rather a best-practice order inspired by our study.

1. Adopt a context-aware integration, as soon as the method covers several structures.
2. Use the certificate-based method, if the underlying problem allows it.
3. If no huge discrepancy in run-time is expected, use derived rules, otherwise reflection. We can approximate such a discrepancy by the inherent size of the proofs and by the amount of computation they contain. For instance qe. has inherently large proofs, containing not much computations.
4. For engineering-driven applications, use reflection in order to scale up, hopefully.

Problem	§	Method
dlo (Langford)	§2.1.3	Context-aware
Simple Sums	§2.2.1	Reflection
Primes	§2.3	Certificates
$\forall$ and $\forall\exists$ in Rings	§3.2.1 and §3.2.2	Certificates and Context-aware
$\forall$ Polynomials in $\mathbb{R}$	§3.3	Certificates
$dlo_{\emptyset}$ (Ferrante and Rackoff)	§4.2	Context-aware
$\mathcal{F}_+^{\rho}$ (parametric ordered fields)	§4.3.1	Reflection
$\mathcal{Z}_+$ (Presburger arithmetic)	§4.3.2	Reflection
$\mathcal{R}_{\lfloor\cdot\rfloor}$ (mixed $\mathbb{R}$ and $\mathbb{Z}$)	§4.3.3	Reflection
ACF	§4.4	Derived rule
RCF	§4.5	Derived rule and Reflection

Figure 5.1: All procedures presented in this thesis

Bibliography

[AF04] Andrew W. Appel and Amy P. Felty. Dependent types ensure partial correctness of theorem provers. 14(1):3–19, 2004.

[Art27] Emil Artin. Über die Zerlegung definiter Funktionen in Quadrate. *Hamburger Abhandlungen*, 5:100–115, 1927.

[Art98] Emil Artin. *Galois theory*. Dover Publications Inc., Mineola, NY, second edition, 1998. Edited and with a supplemental chapter by Arthur N. Milgram.

[AW00] Hirokazu Anai and Volker Weispfenning. Deciding linear-trigonometric problems. In *ISSAC*, pages 14–22, 2000.

[Bal99] Clemens Ballarin. *Computer Algebra and Theorem Proving*. PhD thesis, University of Cambridge, Computer Laboratory, Cambridge, UK, 1999.

[Bal04] Clemens Ballarin. Locales and locale expressions in Isabelle/Isar. In Stefano Berardi et al., editors, *Types for Proofs and Programs (TYPES 2003)*, volume 3085 of *Lecture Notes in Computer Science*, 2004.

[Bal06] Clemens Ballarin. Interpretation of locales in Isabelle: Theories and proof contexts. In J. M. Borwein and W. M. Farmer, editors, *Mathematical Knowledge Management (MKM 2006)*, volume 4108 of *Lecture Notes in Artificial Intelligence*, 2006.

[Bar00] Bruno Barras. Programming and computing in HOL. In *Proceedings of the 13th International Conference on Theorem Proving in Higher Order Logics*, Lecture Notes in Computer Science, pages 17–37. Springer-Verlag, 2000.

[BB02] Henk Barendregt and Erik Barendsen. Autarkic computations in formal proofs. *Journal of Functional Programming*, 28(3):321–336, 2002.

[Bel07] Karim Belabas *et al*. *PARI/GP, version* `2.3.1`. Bordeaux, 2007. `http://pari.math.u-bordeaux.fr/`.

[Ber77] Leonard Berman. Precise bounds for Presburger arithmetic and the reals with addition: Preliminary report. In *FOCS*, pages 95–99. IEEE, 1977.

[Ber80] Leonard Berman. The complexity of logical theories. *Theoretical Computer Science*, 11:71–77, 1980.

[BGD03] Sergey Berezin, Vijay Ganesh, and David L. Dill. An online proof-producing decision procedure for mixed-integer linear arithmetic. In Hubert Garavel and John Hatcliff, editors, *TACAS*, volume 2619 of *LNCS*, pages 521–536. Springer, 2003.

[BJW05] Bernard Boigelot, Sébastien Jodogne, and Pierre Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(3):614–633, 2005.

[BM81] R. S. Boyer and J. S. Moore. Metafunctions: Proving them correct and using them efficiently as new proof procedures. In R. S. Boyer and J S. Moore, editors, *The Correctness Problem in Computer Science*, chapter 3, pages 103–184. Academic Press, New York, 1981.

[BN98] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, New York, NY, USA, 1998.

[BN00] Stefan Berghofer and Tobias Nipkow. Executing higher order logic. In *Types for Proofs and Programs (TYPES 2000)*, volume 2277 of *Lecture Notes in Computer Science*, pages 24–40. Springer-Verlag, 2000.

[BOKR84] Michael Ben-Or, Dexter Kozen, and John Reif. The complexity of elementary algebra and geometry. In *STOC '84: Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 457–464, New York, NY, USA, 1984. ACM.

[BPR03] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, 2003.

[BR90] Riccardo Benedetti and Jean Jacques Risler. *Real algebraic and semi-algebraic sets*. Hermann, éditeurs des sciences et des arts, Paris, 1990.

[Bro87] W. Dale Brownawell. Bounds for the degrees in the nullstellensatz. *The Annals of Mathematics, 2nd Ser.*, 126(3):577–591, 1987.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis, Mathematical Institute, University of Innsbruck, Austria, 1965.

[Bün91] R. Bündgen. Simulating Buchberger's algorithm by Knuth-Bendix completion. pages 386 – 397, 1991.

[BW01] Alexander Bockmayr and Volker Weispfenning. Solving numerical constraints. In Robinson and Voronkov [RV01], chapter 12, pages 751–842.

[BWK93] Thomas Becker, Volker Weispfenning, and Heinz Kredel. *Gröbner bases: a computational approach to commutative algebra*. Springer-Verlag, London, UK, 1993.

[CG84] Alexander L. Chistov and Dima Grigoriev. Complexity of quantifier elimination in the theory of algebraically closed fields. In Michal Chytil and Václav Koubek, editors, *MFCS*, volume 176 of *Lecture Notes in Computer Science*, pages 17–31. Springer-Verlag, 1984.

[CG01] S. C. Chou and X. S. Gao. Automated reasoning in geometry. In Robinson and Voronkov [RV01], chapter 11, pages 707–749.

[Cha03] Amine Chaieb. Isabelle trifft Presburger Arithmetik. Master's thesis, TU München, 2003.

[Cha06a] Amine Chaieb. Mechanized quantifier elimination for linear real-arithmetic in Isabelle/HOL. Technical report, TU München, 2006.

[Cha06b] Amine Chaieb. Verifying mixed real-integer quantifier elimination. In Furbach and Shankar [FS06], pages 528–540.

[CLR95] M. D. Choi, T. Y. Lam, and B. Reznick. Sums of squares of real polynomials. In *Symp. in Pure Math.*, volume 58, pages 103–126. AMS, Providence, R.I., 1995.

[CN05] Amine Chaieb and Tobias Nipkow. Verifying and reflecting quantifier elimination for Presburger arithmetic. In G. Sutcliffe and A. Voronkov, editors, *Logic for Prog., Art. Intelligence, and Reasoning*, volume 3835 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2005.

[CN06] Amine Chaieb and Tobias Nipkow. Proof Synthesis and Reflection for Linear Arithmetic. Submitted to JAR, 2006.

[Coh69] P. J. Cohen. Decision procedures for real and p-adic fields. *Communications in Pure and Applied Mathematics*, 22:131–151, 1969.

[Col75] George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In H. Barkhage, editor, *Automata Theory and Formal Languages*, volume 33 of *LNCS*, pages 134–183. Springer, 1975.

[Coo72] D. C. Cooper. Theorem proving in arithmetic without multiplication. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 7, pages 91–100. Edinburgh University Press, 1972.

[Cré04] Pierre Crégut. Une procédure de décision réflexive pour un fragment de l'arithmétique de Presburger. In *Informal proceedings of the 15th journées francophones des langages applicatifs*, 2004.

[CW07] Amine Chaieb and Makarius Wenzel. Context aware calculation and deduction — ring equalities via Gröbner Bases in Isabelle. In Kauers et al. [KKMW07], pages 27–39.

[Dav57] Martin Davis. A computer program for Presburger's algorithm. In *Summaries of talks presented at the Summer Inst. for Symbolic Logic, Cornell University*, pages 215–233. Inst. for Defense Analyses, Princeton, NJ, 1957.

[Del02] David Delahaye. A proof dedicated meta-language. In *Logical Frameworks and Meta-Languages (LFM 2002)*, ENTCS 70(2), 2002.

[DH88] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1/2):29–35, 1988.

[DM06] David Delahaye and Micaela Mayero. Quantifier elimination over algebraically closed fields in a proof assistant using a computer algebra system. *ENTCS*, 151(1):57–73, 2006.

[DS97] Andreas Dolzmann and Thomas Sturm. REDLOG: computer algebra meets computer logic. *SIGSAM Bulltin*, 31(2):2–9, 1997.

[DSW98a] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. A new approach for automatic theorem proving in real geometry. *Journal of Functional Programming*, 21(3):357–380, 1998.

[DSW98b] Andreas Dolzmann, Thomas Sturm, and Volker Weispfenning. Real quantifier elimination in practice. In B. H. Matzat, G.-M. Greuel, and G. Hiss, editors, *Algorithmic Algebra and Number Theory*, pages 221–247. Springer-Verlag, 1998. Also as tech. rep. MIP-9720 of Universität Passau 1997.

[Ebb04] H-D. Ebbinghaus *et al*. *Zahlen*. Springer-Verlag, Germany, 3rd. edition, 2004.

[Est56] T. Estermann. On the fundamental theorem of algebra. *Journal of the London Mathematical Society*, 31:238–240, 1956.

[Far07] William M. Farmer. Biform theories in Chiron. In Kauers et al. [KKMW07], pages 66–79.

[FR74] Michael Fischer and Michael Rabin. Super-exponential complexity of Presburger arithmetic. In *SIAMAMS: Complexity of Computation: Proceedings of a Symposium in Applied Mathematics of the American Mathematical Society and the Society for Industrial and Applied Mathematics*, 1974.

[FR75] Jeanne Ferrante and Charles Rackoff. A decision procedure for the first order theory of real addition with order. *SIAM J. of Computing*, 4(1):69–76, 1975.

[FR79] Jeanne Ferrante and Charles Rackoff. *The Computational Complexity of Logical Theories*, volume 718 of *Lecture Notes in Mathematics*. Springer-Verlag, NY, 1979.

[FS06] Ulrich Furbach and Natarajan Shankar, editors. *Automated Reasoning, Third International Joint Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006.

[GBD02] Vijay Ganesh, Sergey Berezin, and David L. Dill. Deciding presburger arithmetic by model checking and comparisons with other methods. In Mark Aagaard and John W. O'Leary, editors, *FMCAD*, volume 2517 of *LNCS*, pages 171–186. Springer, 2002.

[GM93] M. J. C. Gordon and T. F. Melham, editors. *Introduction to HOL: a theorem proving environment for higher order logic*. Cambridge University Press, New York, NY, USA, 1993.

[GM05] Benjamin Grégoire and Assia Mahboubi. Proving equalities in a commutative ring done right in coq. In Hurd and Melham [HM05], pages 98–113.

[GMM+78] M. Gordon, R. Milner, L. Morris, M. Newey, and C. Wadsworth. A metalanguage for interactive proof in LCF. In *POPL '78: Proceedings of the 5th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 119–130, New York, NY, USA, 1978. ACM Press.

[GMW79] M. Gordon, R. Milner, and C. P. Wadsworth. *Edinburgh LCF: A Mechanized Logic of Computation*. Number 78 in LNCS. Springer-Verlag, 1979.

[Gri88] Dima Grigoriev. Complexity of deciding Tarski algebra. *Journal of Symbolic Computation*, 5(1-2):65–108, 1988.

[GT06] Benjamin Grégoire and Laurent Théry. A purely functional library for modular arithmetic and its application for certifying large prime numbers. In Furbach and Shankar [FS06], pages 423–437.

[GTW06] Benjamin Grégoire, Laurent Théry, and Benjamin Werner. A computational approach to Pocklington certificates in type theory. In M. Hagiya and P. Wadler, editors, *Proceedings of FLOPS'06*, volume 3945 of *Lecture Notes in Computer Science*, pages 97 – 113. Springer-Verlag, 2006.

[GV88] Dima Grigoriev and Nicolai Vorobjov. Solving systems of polynomial inequalities in subexponential time. *Journal of Symbolic Computation*, 5(1-2):37–64, 1988.

[Har95] John Harrison. Metatheory and reflection in theorem proving: A survey and critique. Technical Report CRC-053, SRI Cambridge, Millers Yard, Cambridge, UK, 1995.

[Har96] John Harrison. *Theorem proving with the real numbers*. PhD thesis, University of Cambridge, Computer Laboratory, 1996.

[Har01] John Harrison. Complex quantifier elimination in HOL. In Richard J. Boulton and Paul B. Jackson, editors, *TPHOLs 2001: Supplemental Proceedings*, pages 159–174. Division of Informatics, University of Edinburgh, 2001.

[Har06] John Harrison. *HOL Light Tutorial (for version 2.20)*, September 2006.

[Har07a] John Harrison. Automating elementary number-theoretic proofs using Gröbner bases. In Frank Pfenning, editor, *Proceedings of CADE 21*, volume 4603 of *Lecture Notes in Computer Science*, pages 51–66, Bremen, Germany, 2007. Springer-Verlag.

[Har07b] John Harrison. Verifying nonlinear real formulas via sums of squares. In Klaus Schneider and Jens Brandt, editors, *Proceedings of the 20th International Conference on Theorem Proving in Higher Order Logics TPHOLs 2007*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118. Springer-Verlag, 2007.

[Har08] John Harrison. *Introduction to Logic and Theorem Proving*. Cambridge University Press, 2008. *might appear under an different title*.

[HB68] David Hilbert and Paul Bernays. *Grundlagen der Mathematik I*. Springer-Verlag, 1968. 2. Auflage.

[Hei83] Joos Heintz. Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 24:239–277, 1983.

[Hei85] Joos Heintz. Corrigendum: Definability and fast quantifier elimination in algebraically closed fields. *Theoretical Computer Science*, 39:343, 1985.

[Her26] Grete Hermann. Die Frage der endlich vielen Schritte in der Theorie der Polynomideale. *Mathematische Annalen*, 95:736–788, 1926.

[Hil88] David Hilbert. Über die Darstellung definiter Formen als Summe von Formenquadraten. *Mathematische Annalen*, 32:342–350, 1888.

[Hil90] David Hilbert. Über die Theorie der algebraischen Formen. *Mathematische Annalen*, 36:473–534, 1890.

[Hil93] David Hilbert. Über die vollen Invariantensysteme. *Mathematische Annalen*, 42:313–373, 1893.

[HM05] Joe Hurd and Thomas F. Melham, editors. *Theorem Proving in Higher Order Logics, 18th International Conference, TPHOLs 2005, Oxford, UK, August 22-25, 2005, Proceedings*, volume 3603 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.

[HN07] Florian Haftmann and Tobias Nipkow. A code generator framework for Isabelle/HOL. In K. Schneider and J. Brandt, editors, *Theorem Proving in Higher Order Logics: Emerging Trends*. CS Department, University of Kaiserslautern, 2007.

[Hör83] Lars Hörmander. *The Analysis of Linear Partial Differential Operators II*, volume 275 of *Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, 1983.

[HSA06] John Harrison, Konrad Slind, and Rob Arthan. HOL. In Wiedijk [Wie06].

[HT98] John Harrison and Laurent Théry. A skeptic's approach to combining HOL and Maple. *Journal of Functional Programming*, 21:279–294, 1998.

[Huy86] Dung T. Huynh. A superexponential lower bound for gröbner bases and church-rosser commutative thue systems. *Information and Control*, 68(1-3):196–206, 1986.

[HW75] J. Heintz and R. Wüthrich. An efficient quantifier elimination algorithm for algebraically closed fields of any characteristic. *SIGSAM Bull.*, 9(4):11–11, 1975.

[HW06] Florian Haftmann and Makarius Wenzel. Constructive type classes in Isabelle. In Thorsten Altenkirch and Conor McBride, editors, *Types for Proofs and Programs*, 2006.

[Jac06] Paul Jackson. Nuprl. In Wiedijk [Wie06].

[Kap96] Deepak Kapur. Automated geometric reasoning: Dixon resultants, Gröbner bases, and characteristic sets. In Dongming Wang, editor, *Automated Deduction in Geometry, International Workshop on Automated Deduction in Geometry*, volume 1360 of *Lecture Notes in Computer Science*, pages 1–36. Springer, 1996.

[KK67] G. Kreisel and J.L. Krivine. *Elements of mathematical logic (Model theory)*. North-Holland, Amsterdam, 1967.

[KKMW07] Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors. *Towards Mechanized Mathematical Assistants, 14th Symposium, Calculemus 2007, 6th International Conference, MKM 2007, Hagenberg, Austria, June 27-30, 2007, Proceedings*, volume 4573 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.

[Kla04] Felix Klaedtke. On the automata size for Presburger arithmetic. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS 2004)*, pages 110–119. IEEE Computer Society Press, 2004.

[KM97] Matt Kaufmann and J S. Moore. An industrial strength theorem prover for a logic based on common Lisp. *IEEE Transactions on Software Engineering*, 23(4):203–213, 1997.

[Kol88] J. Kollár. Sharp effective nullstellensatz. *Journal of the AMS*, 1:963–975, 1988.

[Kra06] Alexander Krauss. Partial recursive functions in higher-order logic. In Furbach and Shankar [FS06], pages 589–603.

[KS04] R. Klapper and A. Stump. Validated Proof-Producing Decision Procedures. In C. Tinelli and S. Ranise, editors, *2nd Int. Workshop Pragmatics of Decision Procedures in Automated Reasoning*, 2004.

[KT31] Casimir Kuratowski and Alfred Tarski. Les opérations logiques et les ensembles projectifs. *Fundamentae Mathematicae*, pages 240–248, 1931.

[KW07] Cezary Kaliszyk and Freek Wiedijk. Certified computer algebra on top of an interactive theorem prover. In Kauers et al. [KKMW07], pages 94–105.

[KWP99] Florian Kammüller, Makarius Wenzel, and Lawrence C. Paulson. Locales: A sectioning concept for Isabelle. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, *Theorem Proving in Higher Order Logics (TPHOLs '99)*, volume 1690 of *LNCS*, 1999.

[Lan27] C. H. Langford. Some theorems on deducibility (2nd series). *Annals of mathematics*, 28:16–40, 1927.

[Lit41] J. E. Littlewood. Mathematical notes (14): "Every polynomial has a root". *Journal of the London Mathematical Society*, 16:95–98, 1941.

[Löw15] Leopold Löwenheim. Über Möglichkeiten im logischen Kalkül. *Mathematische Annalen*, 76:447–470, 1915.

[LW93] Rüdiger Loos and Volker Weispfenning. Applying linear quantifier elimination. *Computer Journal*, 36(5):450–462, 1993.

[Mah06a] Assia Mahboubi. *Contributions à la certification des calculs sur $\mathbb{R}$: théorie, preuves, programmation*. PhD thesis, Univ. Nice Sophia-Antipolis, 2006.

[Mah06b] Assia Mahboubi. Proving formally the implementation of an efficient gcd algorithm for polynomials. In Furbach and Shankar [FS06], pages 438–452.

[May97] Ernst W. Mayr. Some complexity results for polynomial ideals. *Journal of Complexity*, 13(3):303–325, Sep 1997.

[MdR94] M. Mauny and Daniel de Rauglaudre. A complete and realistic implementation of quotations for ML. In *ACM SIGPLAN Workshop on Standard ML and its Applications*, 1994.

[MH05] Sean McLaughlin and John Harrison. A Proof-Producing Decision Procedure for Real Arithmetic. In Robert Nieuwenhuis, editor, *CADE-20: 20th International Conference on Automated Deduction, proceedings*, volume 3632 of *Lecture Notes in Computer Science*, pages 295–314. Springer-Verlag, 2005.

[Mis93] Bhubaneswar Mishra. *Algorithmic algebra*. Springer-Verlag, 1993.

[MM82] Ernst W. Mayr and Albert R. Meyer. The complexity of the word problems for commutative semigroups and polynomial ideals. *Advances in Mathematics*, 45:305–329, 1982.

[MMvdD83] Angus Macintyre, Kenneth McKenna, and Lou van den Dries. Elimination of quantifiers in algebraic structures. *Advances in Mathematics*, 47:74–87, 1983.

[Mon74] Leonard Monk. *An elementary recursive decision procedure for $Th(\mathbb{R}, +, \cdot)$*. PhD thesis, University of California, Berkeley, 1974.

[Nip08] Tobias Nipkow. Reflecting quantifier elimination for linear arithmetic. In K. Spies, editor, *Proceedings Marktoberdorf Summer School 2007*. IOS Press, 2008. To appear.

[Nor03] Michael Norrish. Complete integer decision procedures as derived rules in HOL. In D.A. Basin and B. Wolff, editors, *Theorem Proving in Higher Order Logics, TPHOLs 2003*, volume 2758 of *Lecture Notes in Computer Science*, pages 71–86. Springer-Verlag, 2003.

[NPW02] Tobias Nipkow, Lawrence C. Paulson, and Makarius Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *Lecture Notes in Computer Science*. Springer-Verlag, 2002.

[Obu05] Steven Obua. Proving bounds for real linear programs in isabelle/hol. In Hurd and Melham [HM05], pages 227–244.

[Opp73] D. C. Oppen. Elementary bounds for Presburger arithmetic. In *STOC '73: Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 34–37, New York, NY, USA, 1973. ACM Press.

[Par00] Pablo Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, Pasadena, CA, 2000.

[Par03] Pablo A. Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical Programming*, 96(2):293–320, 2003.

[Pau87] L. C. Paulson. *Logic and Computation*. Cambridge University Press, 1987.

[Pól28] G. Pólya. Über positive Darstellungen von Polynomen. *Vierteljahresschrift der Naturforschenden Gesellschaft in Zürich*, 73:141–145, 1928.

[PP07] H. Peyrl and P. A. Parrilo. A Macaulay 2 package for computing sum of squares decompositions of polynomials with rational coefficients. In *Symbolic-Numeric Computation*, pages 207–208, London, Ontario, Canada, July 2007.

[PPSP04] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo. *SOSTOOLS: Sum of squares optimization toolbox for MATLAB*, 2004.

[Pre29] Mojzesz Presburger. Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In *Comptes rendus du I congrès de math. des pays slaves*, pages 92–101, 1929.

[Pug91] William Pugh. The Omega test: a fast and practical integer programming algorithm for dependence analysis. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 4–13. ACM Press, 1991.

[PW98] Victoria Powers and Thorsten Wörmann. An algorithm for sums of squares of real polynomials. *Journal of Pure and Applied Algebra*, 127:99–104, 1998.

[Ren92a] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, part I: Introduction. preliminaries. the geometry of semi-algebraic sets. the decision problem for the existential theory of the reals. *Journal of Symbolic Computation*, 13(3):255–300, 1992.

[Ren92b] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, part II: The general decision problem. preliminaries for quantifier elimination. *Journal of Symbolic Computation*, 13(3):301–328, 1992.

[Ren92c] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals, part III: Quantifier elimination. *Journal of Symbolic Computation*, 13(3):329–352, 1992.

[Ric74] Fred Richman. Constructive aspects of Noetherian rings. *Proceedings of the AMS*, 44(2):436–441, 1974.

[RL78] C. R. Reddy and D. W. Loveland. Presburger arithmetic with bounded quantifier alternation. In *STOC '78: Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 320–325, New York, NY, USA, 1978. ACM Press.

[Rob49] Julia Robinson. Definability and decision problems in arithmetic. *Journal of Symbolic Logic*, 14:98–114, 1949.

[RV01] A. Robinson and A. Voronkov, editors. *Handbook of Automated Reasoning*, volume I. Elsevier Science, 2001.

[Sch91] Konrad Schmüdgen. The k-moment problem for compact semi-algebraic sets. *Mathematische Annalen*, 289(1):87–97, 1991.

[Sch99] Markus Schweighofer. Algorithmische Beweise für Nichtnegativ- und Positivstellensätze. Master's thesis, Universität Passau, 1999.

[Sei74] Abraham Seidenberg. Constructions in Algebra. *Transactions of the AMS*, 197:273–313, 1974.

[Sko19] Thoralf Skolem. Untersuchungen über die Axiome des Klassenkalküls und über Produktations- und Summationsprobleme, welche gewisse Klassen von Aussagen betreffen. *Skrifter Videnskapsakademiet i Kristiania*, 3:37–71, 1919. Written in 1917, reprinted in [Sko70b].

[Sko28] Thoralf Skolem. Über die mathematische Logik. *Norsk Mathematisk Tidsskrift*, 10:125–142, 1928.

[Sko30] Thoralf Skolem. Über einige Satzfunktionen in der Arithmetik. In *Skrifter utgitt av Det Norske Videnskaps-Akademi i Oslo, I. Matematisk Naturvidenskapelig Klasse*, volume 7, pages 1–28. Oslo, 1930. Reprinted in [Sko70b].

[Sko70a] Thoralf Skolem. Logisch-kombinatorische Untersuchungen über die Erfüllbarkeit und Beweisbarkeit mathematischen Sätze nebst einem Theoreme über dichte Mengen. In *Selected Works in logic* [Sko70b], pages 103–136. Edited by Jens Erik Fenstad.

[Sko70b] Thoralf Skolem. *Selected Works in Logic*. Universitetsforlaget, Oslo-Bergen-Tromsö, 1970. Edited by Jens Erik Fenstad.

[Sli96] Konrad Slind. Function definition in higher-order logic. In *TPHOLs '96: Proceedings of the 9th International Conference on Theorem Proving in Higher Order Logics*, pages 381–397, Turku, Finland, 1996. Springer-Verlag.

[Sli97] Konrad Slind. Derivation and use of induction schemes in higher-order logic. In *TPHOLs '97: Proceedings of the 10th International Conference on Theorem Proving in Higher Order Logics*, pages 275–290, London, UK, 1997. Springer-Verlag.

[Smo91] Craig Smoryński. *Logical Number Theory I, An Introduction*. Springer-Verlag, 1991.

[Ste74] Gilbert Stengle. A Nullstellensatz and a Positivstellensatz in semialgebraic geometry. *Mathematische Annalen*, 207(2):87–97, 1974.

[Tar31] Alfred Tarski. Sur les ensembles définissables de nombres réels. *Fundamentae Mathematicae*, 17:210–239, 1931. Translated as "On definable sets of real numbers", in: J. Corcoran (Ed.), Logic, Semantics, Metamathematics: papers from 1923 to 1938, Hackett Publishing Company, Indianapolis, IN, 1983, pp. 110-142.

[Tar51] Alfred Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[TH07] Laurent Théry and Guillaume Hanrot. Primality proving with elliptic curves. In Klaus Schneider and Jens Brandt, editors, *TPHOLs*, volume 4732 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2007.

[Thé] Laurent Théry. Presburger's algorithm. `http://coq.inria.fr/contribs/Presburger.html`.

[TLG06] Laurent Théry, Pierre Letouzey, and Georges Gonthier. Coq. In Wiedijk [Wie06].

[TM49] Alfred Tarski and Andrzej Mostowski. Arithmetical classes and types of well ordered systems. *Bulletin of the AMS*, 55:65, 1949.

[vdD80] Lou P. D. van den Dries. A linearly ordered ring whose theory admits elimination of quantifiers is a real closed field. *Proceedings of the American Mathematical Society*, 79(1):97–100, Mat 1980.

[WB95] P. Wolper and B. Boigelot. An automata-theoretic approach to presburger arithmetic constraints (extended abstract). In *SAS '95: Proc. of the Second Int. Symp. on Static Analysis*, pages 21–32, London, UK, 1995. Springer-Verlag.

[WC07] Makarius Wenzel and Amine Chaieb. SML with antiquotations embedded into Isabelle/Isar. In Jacques Carette and Freek Wiedijk, editors, *Workshop on Progr. Lang. for Mechanized Math. (part of CALCULEMUS 2007). Hagenberg, Austria*, June 2007.

[Wei83] Volker Weispfenning. Aspects of quantifier elimination in algebra. In *Proc. 25 Universal Algebra and its links with logic, Arbeitstagung Darmstadt*, pages 85–105. Heldermann V., Berlin, 1983.

[Wei88] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1/2):3–27, 1988.

[Wei90] Volker Weispfenning. The complexity of almost linear diophantine problems. *Journal of Symbolic Computation*, 10(5):395–404, 1990.

[Wei94a] Volker Weispfenning. Parametric linear and quadratic optimization by elimination. Technical Report MIP-9404, Universität Passau, Passau, April 1994.

[Wei94b] Volker Weispfenning. Quantifier elimination for real algebra — the cubic case. In *International Symposium on Symbolic and Algebraic Computation*, pages 258–263, 1994.

[Wei97a] Volker Weispfenning. Complexity and uniformity of elimination in Presburger arithmetic. In *ISSAC*, pages 48–53, 1997.

[Wei97b] Volker Weispfenning. Quantifier elimination for real algebra – the quadratic case and beyond. *Applicable Algebra in Engineering, Comm. and Computing*, 8(2):85–101, 1997.

[Wei98] Volker Weispfenning. A new approach to quantifier elimination for real algebra. In B.F. Caviness and J. R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 376–392. Springer-Verlag, 1998.

[Wei99] Volker Weispfenning. Mixed real-integer linear quantifier elimination. In *ISSAC '99*, pages 129–136, New York, NY, USA, 1999. ACM Press.

[Wei00] Volker Weispfenning. Deciding linear-exponential problems. *SIGSAM Bulletin (ACM Special Interest Group on Symbolic and Algebraic Manipulation)*, 34(1):30–31, 2000.

[Wei06] Volker Weispfenning. Private communication at Passau, June 2006.

[Wel95] Morten Welinder. Very efficient conversions. In *Higher Order Logic, Theorem Proving, and Its Applications*, volume 971 of *Lecture Notes in Computer Science*, pages 340–352, 1995.

[Wen97] Makarius Wenzel. Type classes and overloading in higher-order logic. In Elsa L. Gunter and Amy Felty, editors, *Theorem Proving in Higher Order Logics (TPHOLs '97)*, volume 1275 of *LNCS*, 1997.

[Wen02] Makarius Wenzel. *Isabelle/Isar — A Versatile Environment for Human-Readable Formal Proof Documents*. PhD thesis, TU München, 2002. `http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/wenzel.html`.

[Wen07] Makarius Wenzel. *The Isabelle/Isar Reference Manual (for Isabelle2007)*, 2007.

[Wie06] Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2006.

[WP06] Makarius Wenzel and Lawrence C. Paulson. Isabelle/Isar. In Wiedijk [Wie06].

[Wüt77] H. R. Wüthrich. *Ein schnelles Quantoreneliminationsverfahren für die Theorie der algebraisch abgeschlossenen Körper*. PhD thesis, Uiversität Zürich, 1977.

[Wüt74] H. R. Wüthrich. Ein Entscheidungsverfahren für die Theorie der reellabgeschlossenen Körper. In *Komplexität von Entscheidungsproblemen–Ein Seminar*, volume 43 of *Lecture Notes in Computer Science*, pages 138–162. Springer-Verlag, 1973/74.