P-adic Constraint Solving

Andreas Dolzmann and Thomas Sturm
Department of Mathematics and Computer Science
University of Passau, Germany
{dolzmann,sturm}@uni-passau.de
http://www.fmi.uni-passau.de/~{dolzmann,sturm}/

MIP-9901
January 14, 1999

Abstract

We automatically check for the feasibility of arbitrary boolean combinations of linear parametric p-adic constraints using a quantifier elimination method. This can be done uniformly for all p. We focus on the necessary simplification methods. Our method is implemented within the computer algebra system REDUCE. We illustrate the applicability of this implementation to non-trivial problems including the solution of systems of linear congruences over the integers.

1 Introduction

It is well-known that linear parametric constraint solving over the reals has numerous important applications in science and engineering. The same holds for corresponding integer and mixed real-integer problems.

In this article we consider analogue problems over p-adic numbers instead of real numbers. This also has important though less obvious applications, mainly in class field theory and Diophantine analysis [Dub92].

One can, for instance, weaken the problem of finding integer solutions to a Diophantine polynomial equation $f(x_1, \dots, x_n) = 0$ to considering for a fixed prime p only congruences $f(x_1, \dots, x_n) \equiv 0 \pmod{p^k}$ for all prime powers. This can in turn be reduced to considering the initial equation over the p-adic integers, which is much easier than over $\mathbb{Z}$.

An important special case of our linear method is testing for the feasibility and finding sample solutions of a system of simultaneous congruences in linear variables $x_1, \dots, x_n$ over the integers:

$$
\begin{aligned}
a_{11}x_1 + \cdots + a_{1n}x_n &\equiv b_1 \pmod{p^{k_1}} \\
&\vdots \\
a_{m1}x_1 + \cdots + a_{mn}x_n &\equiv b_m \pmod{p^{k_m}}.
\end{aligned}
$$

As a very important feature, our approach can solve problems uniformly for all p-adic valuations. As a consequence, variable-free constraints can, in general, not be evaluated to truth values. Such a constraint might e.g. state that both 1 and 2 have the same value, which is true only for $p \neq 2$. Hence, the quantifier elimination procedure does not amount to a decision procedure when eliminating all the variables. It is crucial to have sophisticated simplification methods at hand to obtain comprehensible results.

Both the quantifier elimination and the simplification methods discussed here are efficiently implemented within the REDUCE package REDLOG. Source code and documentation are freely available.

The plan of the paper is as follows: Section 2 summarizes some basic facts on valued fields, and introduces the formal language we use for our constraints. Section 3 sketches the elimination method, which is described in detail elsewhere [Stu98]. In Section 4 we describe in detail the mathematics behind our simplification method and its algorithmic realization. Section 5 provides a brief summary of the relevant features of REDLOG. Section 6 is a collection of computation examples by our implementation in REDLOG. Section 7 finally summarizes our results.

2 Some facts on valued fields

Given a field K and an ordered additive Abelian group Γ, a *valuation* $v : K \to \Gamma \cup \{\infty\}$ is a map with $v(a) = \infty$ if and only if $a = 0$, and

$$v(ab) = v(a) + v(b), \quad v(a+b) \geq \min(v(a), v(b)).$$

It follows that $v(a+b) = \min(v(a), v(b))$ if $v(a) \neq v(b)$. This fact is referred to as *ultrametric triangle equality.*

Consider e.g. the rational numbers. For any prime $p \in \mathbb{N}$, they allow the p-adic valuation $v_p : \mathbb{Q} \to \mathbb{Z} \cup \{\infty\}$ defined by $v_p(0) = \infty$ and

$$v_p(r/s) = \max\{\, n \in \mathbb{N} : p^n \mid r \,\} - \max\{\, n \in \mathbb{N} : p^n \mid s \,\}$$

for $r/s \in \mathbb{Q}^{\times}$. Note that for $z \in \mathbb{Z}$ we have $v_p(p^z) = z$, i.e., v_p is onto. Due to a famous theorem by Ostrowski [Ost18] the p-adic valuations are essentially the only possible valuations on $\mathbb{Q}$.

The interest for valuations origins in considering possible absolute values for fields. In fact, for $\Gamma \subseteq \mathbb{R}$ absolute values obeying the ultrametric triangle inequality $|a+b| \leq \max(|a|, |b|)$ can be obtained as $|a|_v = 2^{-v(a)}$ and vice versa. For the p-adic valuations one usually replaces $|a|_p = p^{-v_p(a)}$.

Given such an absolute value $|\cdot|_v$, the valued field K can be completed by adding limits for all Cauchy sequences wrt. $|\cdot|_v$ in analogy to the construction of the reals. For any $|\cdot|_p$ on $\mathbb{Q}$ this process yields the well-known p-adic numbers.

One reason for switching from absolute values to maps v as discussed here is that the latter algebraically structure the field K: The elements of non-negative value form a ring, the *valuation ring* R_v. In R_v the elements of positive value form a maximal ideal, the *valuation ideal* I_v, which is the only maximal ideal in R_v. The elements $U_v = R_v \setminus I_v$ form the multiplicative group of units of R_v. From the maximality of I_v it follows that $K_v = R_v/I_v$ is a field, the *residue class field* wrt. v. All ideals in R_v are of the form

$$I_\gamma = \{\, a \in R_v : v(a) > \gamma \,\} \quad \text{for} \quad 0 \leq \gamma \in \Gamma.$$

The situation is pictured in Figure 1.

The image $v(K^{\times}) \subseteq \Gamma$ of the multiplicative group of K is called the *valuation group* of K and v. The valuation is called discrete if this valuation group is discrete, i.e. contains an element of minimal positive value. The p-adic valuations are discrete with value group $\mathbb{Z}$.

A valuation can be essentially recovered from its valuation ring R_v possibly switching to an isomorphic valuation group. To avoid a two-sorted

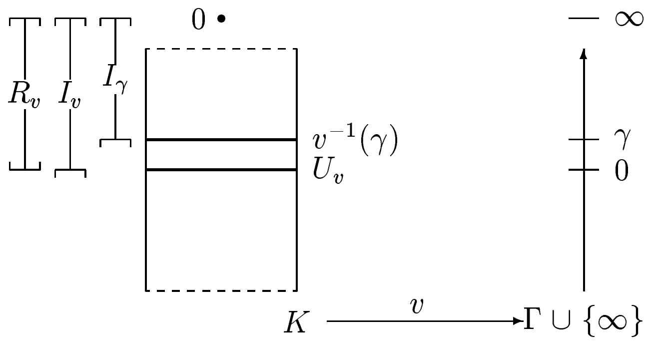

Figure 1: The structure of a valued field

language, we may thus drop the information about the actual value group by using the language of rings together with abstract divisibilities. These divisibilities express ordering relations in the value group by relating field elements:

$$\begin{aligned} x \mid y &\longleftrightarrow v(x) \leq v(y), \\ x \parallel y &\longleftrightarrow v(x) < v(y), \\ x \sim y &\longleftrightarrow v(x) = v(y). \end{aligned}$$

Note that $\neg(x \mid y) \longleftrightarrow y \parallel x$ and vice versa. For convenience, we also introduce $x \not\sim y \longleftrightarrow \neg(x \sim y)$ and write $x \neq y$ for $\neg(x = y)$. For any constraint γ we denote by $\overline{\gamma}$ the constraint equivalent to $\neg\gamma$. For discretely valued fields, we furthermore add a constant π of value 1 to our language.

Note that our language does not include reciprocals. For convenience, we allow ourselves to identify terms with polynomials in $\mathbb{Z}[\underline{X}, \pi]$ where the set $\underline{X}$ stands for the contained variables, and π is the constant of our language.

3 An outline of the method

For solving our linear constraints, we use an effective linear quantifier elimination procedure based on *virtual substitution of test points.* Virtual substitution methods date back to a theoretical paper by Weispfenning [Wei88]. Corresponding methods over the reals have been successfully used for solving problems from numerous areas in science and engineering [DSW98].

For eliminating the quantifiers from an input formula

$$\varphi(u_1, \ldots, u_m) \equiv \mathsf{Q}_1 x_1 \ldots \mathsf{Q}_n x_n \psi(u_1, \ldots, u_m, x_1, \ldots, x_n)$$

where $\mathsf{Q}_i \in \{\exists, \forall\}$, the elimination starts with the innermost quantifier regarding the other quantified variables within ψ as extra parameters. Universal quantifiers are handled by means of the equivalence $\forall x \psi \longleftrightarrow \neg \exists x \neg \psi$. We may thus restrict our attention to a formula of the form

$$\varphi^*(u_1, \dots, u_k) \equiv \exists x \psi^*(u_1, \dots, u_k, x),$$

where the $u_{m+1}, \dots, u_k$ are actually x_i quantified from further outside. The idea is now to find a finite *elimination set* E of terms in $u_1, \dots, u_k$ such that

$$\exists x \psi^*(u_1, \dots, u_k, x) \equiv \bigvee_{t \in E} \psi^*[x/t](u_1, \dots, u_k).$$

That is, the above disjunction is a quantifier-free equivalent for φ^*. Note that it is not necessary to perform any transformation on the boolean structure of ψ^*. The elimination method is single exponential in the number of quantified variables, and double exponential in the number of quantifier blocks. It has turned out suitable for parallelization [DGS98].

By keeping track of the terms t substituted during the elimination process, we obtain instead of a quantifier-free equivalent $\bigvee_{i=1}^{k} \psi^*[x/t_i]$ a guarded expression [DS97a]

$$\begin{bmatrix} \psi^*[x/t_1] & x = t_1 \\ \vdots & \vdots \\ \psi^*[x/t_k] & x = t_k \end{bmatrix}$$

including satisfying sample points. This process of *extended quantifier elimination* can also be repeated for several existential quantifiers. The result then is a set of conditions each associated with an answer for each eliminated variable obtained by resubstitution.

The construction of elimination sets for linear formulas in valued fields has been described by the second author [Stu98]. Before, Weispfenning had given elimination sets for special cases of valued fields including the case of p-adic valuations [Wei88].

The existence of a quantifier elimination procedure for the general case including non-linear formulas has been shown independently by Ax and Kochen [AK66] and Ershov [Ers65]. The first explicit procedure has been given by Cohen [Coh69]. Considerable progress has been made by Macintyre [Mac76] turning to a more reasonable language including root predicates in analogy to the reals. This has been made explicit by Weispfenning [Wei84].

Our elimination procedure for linear formulas only relies on elementary arithmetic and on the valuation axioms. In particular, it does not require the considered field to be complete or even Henselian. It follows that our elimination results are correct over both $\mathbb{Q}$ and $\mathbb{Q}_p$. The same applies to our simplification strategies discussed in the following section.

4 Simplification

With virtual substitution methods it has turned out crucial to have sophisticated simplification methods at hand [DS97c]. In fact, the success of REDLOG is based mainly on its powerful simplifier.

The corresponding algorithms have been described in detail by the authors taking ordered fields as an example [DS97c]. We adapt this framework to the theory of valued fields. After defining appropriate *simplification goals*, the task of simplification splits into two major subproblems. Firstly, one has to simplify single constraints. Secondly, one has to simplify nested boolean combinations of constraints detecting algebraic relationships among them.

4.1 Simplification goals

It is not always clear what kind of formulas are to be considered "simple." We summarize and explain our simplification goals:

Few constraints This is clearly one of the main goals. Quantifier elimination output is often too large to be read and understood by a human.

Few different constraints This is very convenient for quantifier elimination by virtual substitution, which we use. We shall see that caring for the identification of equivalent constraints will support our smart simplification, which is concerned with detecting algebraic relationships among separate constraints.

Simple terms This keeps the output small and comprehensible. We prefer a logic representation of knowledge to an algebraic one even for the price of more constraints. For instance, $f = 0 \vee g = 0$ is better than $fg = 0$.

Convenient relations We prefer field constraints to valuation constraints. The former are much more familiar.

Some of these goals obviously contradict one another. They give, however, an idea of the issues addressed by our work.

4.2 Simplification of atomic constraints

The simplification strategies for atomic constraints distinguish between field constraints $f = g$, $f \neq g$ on one hand and valuation constraints $f \mid g$, $f \parallel g$, $f \sim g$, $f \not\sim g$ on the other hand.

We tacitly assume that straightforward simplifications like evaluating constraints with equal sides or lexicographically ordering constraints with symmetric relations such as "$\sim$" are applied whenever appropriate.

4.2.1 Field constraints

The simplification of field constraints has been thoroughly investigated in the context of ordered fields [DS97c]. The following lemma involves only computations that can be performed very efficient. Recall that the computation $f/\gcd(f, f')$ of the squarefree part of a univariate polynomial f can be recursively extended to the multivariate case.

Lemma 1 Consider a p-adic valuation. Let f, $g \in \mathbb{Z}[\underline{X}, \pi]$ with $f \neq g$. Consider a constraint $\varphi \equiv f \mathrel{\varrho} g$ for $\varrho \in \{=, \neq\}$. Denote by h the squarefree part of $f - g$, which is made primitive in such a way that the head coefficient is positive. Then $\varphi \longleftrightarrow h \mathrel{\varrho} 0$. Furthermore, if $h = \pi^n h'$, then we may even set $\varphi \longleftrightarrow h' \mathrel{\varrho} 0$. $\square$

If the left hand side of the resulting constraint is 1, it can finally be evaluated to either "true" or "false."

4.2.2 Valuation constraints

Our first result provides an algorithmic test for replacing within valuation constraints polynomials in our constant π by single monomials. This applies also to field constraints with zero right hand sides since $f = 0 \longleftrightarrow f \sim 0$ and $f \neq 0 \longleftrightarrow f \not\sim 0$.

Lemma 2 Consider wrt. a p-adic valuation

$$f = \sum_{i=0}^{m} z_i \pi^i, \quad z_i \in \mathbb{Z},$$

and let $0 \leq k \leq m$ be minimal with $z_k \neq 0$. Compute the prime factor decomposition $z_k = q_1^{e_1} \cdots q_n^{e_n}$. Assume that for all prime factors q_i with $1 \leq i \leq n$ and for all $k + 1 \leq l \leq m$ with $k + e_i \geq l$ we have $q_i^{1+k+e_i-l} \mid z_l$ over the integers. Then $v_p(f) = v_p(z_k \pi^k)$.

Proof Let $k + 1 \leq l \leq m$. If $p = q_i$, then we have for $k + e_i < l$ that

$$v_p(z_k \pi^k) = e_i + k < l \leq v_p(z_l \pi^l),$$

while for $k + e_i \geq l$ it follows that

$$v_p(z_k \pi^k) = e_i + k < (1 + k + e_i - l) + l \leq v_p(z_l \pi^l).$$

If $p \neq q_i$ for all $1 \leq i \leq n$, we have

$$v_p(z_k \pi^k) = k < l \leq v_p(z_l \pi^l).$$

So for all p-adic valuations we have $v_p(z_k \pi^k) < v_p(z_l \pi^l)$, and we can apply the ultrametric triangle equality. $\square$

The following lemma is concerned with cancelling greatest common divisors from both sides of a constraint.

Lemma 3 Consider a p-adic valuation. Let f, $g \in \mathbb{Z}[\underline{X}, \pi]$ with $f \neq 0$ or $g \neq 0$. Let $h = \gcd(f, g)$, $f_h = f/h$, and $g_h = g/h$. Then

$$\begin{aligned} f \parallel g &\longleftrightarrow f_h \parallel g_h \wedge h \neq 0, \\ f \mid g &\longleftrightarrow f_h \mid g_h \vee h = 0, \\ f \sim g &\longleftrightarrow f_h \sim g_h \vee h = 0, \\ f \not\sim g &\longleftrightarrow f_h \not\sim g_h \wedge h \neq 0. \quad \square \end{aligned}$$

The GCD computation here involves both the polynomial GCD of f and g over $\mathbb{Z}[\underline{X}, \pi]$ and the integer GCD of the corresponding contents. The conditions $h = 0$ and $h \neq 0$ will undergo the field constraint simplification described above.

Since $\gcd(f, 0) = f$, we obtain the following corollary as a special case of Lemma 3.

Corollary 4 Consider a p-adic valuation. Let $f \in \mathbb{Z}[\underline{X}, \pi]$ with $f \neq 0$. Then

$$0 \parallel f \longleftrightarrow \text{false},$$

$$f \mid 0 \longleftrightarrow \text{true},$$

$$f \parallel 0 \longleftrightarrow 0 \not\sim f \longleftrightarrow f \not\sim 0 \longleftrightarrow f \neq 0,$$

$$0 \mid f \longleftrightarrow 0 \sim f \longleftrightarrow f \sim 0 \longleftrightarrow f = 0. \quad \square$$

The next lemma shows how to shift valuation information onto one side of a constraint provided that the valuation is fixed.

Lemma 5 Consider a p-adic valuation. Let f, $g \in \mathbb{Z}[\underline{X}]$ with $f, g \neq 0$. Split both f and g into content and primitive part: $f = c_f \cdot f^*$ and $g = c_g \cdot g^*$, and set $\delta = |v(c_f) - v(c_g)|$. Then for $\varrho \in \{|, \|, \sim, \not\sim\}$ we have

$$f \;\varrho\; g \longleftrightarrow f^* \;\varrho\; p^\delta g^*$$

for $v(c_f) \leq v(c_g)$. Similarly, for $v(c_f) > v(c_g)$, we obtain $f \;\varrho\; g \longleftrightarrow p^\delta f^* \;\varrho\; g^*$. $\square$

According to Lemma 3, within constraints involving only monomial occurrences of π, the occurrence will be restricted to one side of the constraint. One can then use the following simplification to completely get rid of the constant π.

Lemma 6 Consider a p-adic valuation. Let z, $z' \in \mathbb{Z}$ be relatively prime with z, $z' \neq 0$, and let $z = q_1^{e_1} \cdots q_n^{e_n}$ be the prime factor decomposition of z. Let $n \in \mathbb{N}$, and set

$$z_= = \prod\{\, q_i : e_i = n \,\}, \quad z_> = \prod\{\, q_i : e_i > n \,\},$$

and $z_\geq = \prod\{\, q_i : e_i \geq n \,\}$. Then

1. $z'\pi^n \sim z \longleftrightarrow z_= \not\sim 1$ and $z'\pi^n \not\sim z \longleftrightarrow z_= \sim 1$,
2. $z'\pi^n \mid z \longleftrightarrow z_\geq \not\sim 1$ and $z \parallel z'\pi^n \longleftrightarrow z_\geq \sim 1$,
3. $z'\pi^n \parallel z \longleftrightarrow z_> \not\sim 1$ and $z \mid z'\pi^n \longleftrightarrow z_> \sim 1$.

Proof We prove part 1, the other parts are analogous. Consider $z'\pi^n \sim z$. If p does not divide z' we may obviously drop it. If p divides z', then it does not divide z, since z and z' are relatively prime. That is $z \sim 1$, and $\pi^n \sim z$ is "false" as well as $z'\pi^n \sim z$. We thus may also drop z' in this case.

Assume now that $\pi^n \sim z$ holds. Then p occurs with the power of n in z. It follows that p divides $z_=$, and thus $z_= \not\sim 1$. Assume vice versa that $z_= \not\sim 1$. Then p divides $z_=$, and by the definition of $z_=$ we know that p occurs in z with the power of n, i.e, $\pi^n \sim z$.

The second equivalence $z'\pi^n \not\sim z \longleftrightarrow z_= \sim 1$ is simply the negation of the first one. $\square$

Constraints involving only integers can be normalized in such a way that their right hand side is 1.

Lemma 7 Let z, $z' \in \mathbb{Z}$ be relatively prime. Then

1. $z \sim z' \longleftrightarrow zz' \sim 1$ and $z \not\sim z' \longleftrightarrow zz' \not\sim 1$,
2. $z \mid z' \longleftrightarrow z \sim 1$ and $z \parallel z' \longleftrightarrow z' \not\sim 1$. $\square$

The remaining integers in the pure integer constraints can be replaced by their squarefree parts.

Lemma 8 Let z, $z' \in \mathbb{Z}$ be relatively prime. Denote by z_s and z'_s the squarefree parts of z and z', respectively. Then for $\varrho \in \{\mid, \parallel, \sim, \not\sim\}$ we have $z \mathrel{\varrho} z' \longleftrightarrow z_s \mathrel{\varrho} z'_s$. $\square$

For solving systems of integer congruences we will use some extra simplification, which obviously contradicts the goal of few constraints. Instead, it focuses on producing few different constraints:

Lemma 9 Let $z \in \mathbb{Z}$, $z \neq 0$ with prime factor decomposition $z = q_1^{e_1} \cdots q_n^{e_n}$. Then

$$z \sim 1 \longleftrightarrow q_1 \sim 1 \wedge \cdots \wedge q_n \sim 1,$$

and $z \not\sim 1 \longleftrightarrow q_1 \not\sim 1 \vee \cdots \vee q_n \not\sim 1$. $\square$

4.3 Smart simplification

Our simplifier will, of course, perform the following straightforward simplifications: in conjunctions we drop "true," and we kill all other constituents if "false" occurs. For disjunctions the dual simplifications are applied.

Moreover we wish to take into account algebraic relationships between several constraints. For instance, $a = 0 \wedge a \parallel 1$ should become "false." Furthermore, we want to make use of such relationships even for constraints that occur at different places deeply nested inside a complex formula. For this, we construct implicit theories during the recursive elimination process.

A *theory* Θ is a set of constraints considered conjunctive. We say that two formulas $\varphi_1(\underline{X})$ and $\varphi_2(\underline{X})$ are *equivalent wrt.* Θ if $\varphi_1(\underline{a}) \longleftrightarrow \varphi_2(\underline{a})$ for all values a that satisfy Θ. Formally considering our constraint variables as constants in the sense of first-order logic, we may then write

$$\Theta \models (\varphi_1 \longleftrightarrow \varphi_2).$$

Such theories can be passed to our simplifier in order to represent some external knowledge. The crucial point, however, is that they are implicitly constructed by the simplifier itself during recursion. We shall first clarify how to simplify a single conjunction or disjunction wrt. a given theory, and then sketch the implicit theory construction during deep simplification.

4.3.1 Evaluating algebraic relationships

The simplification of a disjunction $\bigvee_i \gamma_i$ wrt. a theory Θ is performed in two steps: First simplify $\bigwedge_i \overline{\gamma_i}$, which is equivalent to $\neg \bigvee_i \gamma_i$, wrt. Θ, and then negate the result back in the same manner. It thus suffices to explain how to simplify a conjunction wrt. Θ.

The simplification of a conjunction $\bigwedge_i \gamma_i$ wrt. a theory Θ is based on the following idea: Regard Θ as a simplified knowledge base. Successively add to Θ each piece of knowledge γ_i keeping Θ simplified. Finally extract from Θ all new knowledge. This new knowledge replaces the input conjunction. Note that any list of constraints input as a theory can be turned into a simplified knowledge base by successively adding the constraints to the empty base.

We have to clarify the following two issues:

1. How do we recognize the new knowledge in the end?

2. How can one add one γ_i to Θ such that the latter remains simplified?

For recognizing the new knowledge there is each constraint in Θ labeled with numbers ranging from 0 to n. Each newly added constraint origined by some γ_i will be then labeled $n+1$. Note that there is in general not simply γ_i added to Θ but only the extra information contained in γ_i. If, e.g., $(x \not\sim y, n) \in \Theta$ and $\gamma_i \equiv x \mid y$, then we will add $(x \parallel y, n+1)$ to Θ and delete $(x \not\sim y, n)$ from Θ. This new constraint $x \parallel y$ is exactly the information we wish to have in the simplified constraint. It is finally extracted due to its label $n + 1$.

Let us now examine in detail how to add γ_i to Θ depending on what is already present there. We successively pick the γ_i, and run through Θ comparing γ_i to each $\vartheta \in \Theta$. The relationships we consider between ϑ and γ_i are that there occur the same terms with different relations. We illustrate the four types of simplification that can occur by example. The complete set of simplifications is collected in Table 1 and Table 2:

1. If $\vartheta \equiv f = 0$ and $\gamma_i \equiv f \neq 0$, then we have discovered an inconsistency. We may abort the computation, and return "false." This case is marked in the tables by "I" for *inconsistent*.

2. If $\vartheta \equiv f \parallel g$ and $\gamma_i \equiv f \neq g$, then $\vartheta \longrightarrow \gamma_i$. We may simply drop γ_i, and proceed to γ_{i+1}. This case is marked by "D" for *drop*.

3. If $\vartheta \equiv f \neq g$ and $\gamma_i \equiv f \parallel g$, then similarly to the previous case, $f \neq g$ can be deleted. This time deletion take place in the theory, and we have to continue scanning Θ with γ_i. This case is marked in the tables as "K" for *kill*.

4. If $\vartheta \equiv f \mid g$ and $\gamma_i \equiv g \mid f$, then $\vartheta \wedge \gamma_i \longleftrightarrow f \sim g$. We thus *change* γ_i into $f \sim g$, kill ϑ, and continue scanning Θ with the new γ_i. One easily verifies by inspection of the tables that with our simplifications it is not necessary to restart scanning Θ with the new γ_i. This case is marked in the tables by "C" together with the new γ_i.

If none of the first two cases happens, the possibly changed γ_i is finally labeled and added to Θ. Recall from the previous section that equations and

Table 1: Smart simplification for matching terms.

				γ_i			
		$=$	$\neq$	$\mid$	$\parallel$	$\sim$	$\not\sim$
	$=$	D	I	D	I	D	I
	$\neq$	I	D	–	K	–	K
ϑ	$\mid$	K	–	D	K	K	C ($\parallel$)
	$\parallel$	I	D	D	D	I	D
	$\sim$	K	–	D	I	D	I
	$\not\sim$	I	D	C ($\parallel$)	K	I	D

Table 2: Smart simplification for crossed terms.

				$\gamma_i \equiv g\ \varrho\ f$	
		$\mid$	$\parallel$	$\sim$	$\not\sim$
	$\mid$	C ($f \sim g$)	I	K	C ($f \parallel g$)
$\vartheta \equiv f\ \sigma\ g$	$\parallel$	I	I	I	D
	$\sim$	D	I		
	$\not\sim$	C ($g \parallel f$)	K		

inequalities are simplified such that their right hand side becomes zero. For the purpose of the simplifications described here, we are, of course, able to match, e.g., $f - g = 0$ with $f \mid g$ by means of a simple subtraction.

4.3.2 Deep simplification

As already indicated, we use our concept of a theory for relating information located on different boolean levels. More precisely, we use constraints, which are located on a certain boolean level, deeper inside the formula by enlarging an implicit theory which is recursively passed down. This technique of *theory inheritance* has been described in detail for ordered fields by the authors [DS97c]. A careful analysis has shown that there are no adaptions necessary for the particular situation of valued fields.

5 Implementation in Redlog

Quantifier elimination and extended quantifier elimination by virtual substitution have been generically implemented within the REDUCE package REDLOG. REDLOG is a *computer logic system* providing not only quantifier elimination but a sophisticated working environment for first-order logic over various languages and theories [DS97b]. The REDLOG source code and documentation are freely available on the WWW.[1]

After the great success of REDLOG for solving real problems, quantifier elimination and extended quantifier elimination for p-adic valued fields have been added as instances of the generic elimination code.

The corresponding REDLOG context `dvfsf` (see [DS97b] for details) corresponds to p-adic valued fields. When switching to it, one passes a parameter q which is a possibly negative prime or 0. For positive q all computations take place wrt. the corresponding q-adic valuation. For $q = 0$, the computations are uniformly correct for all p-adic valuations. Both input and output possibly involve a symbolic constant "p" which corresponds to the constant π of our language. Finally, for negative q, the "$-$" can be read as "up to," i.e., all computations are performed in such a way that they are correct for all p-adic valuations with $p \leq |q|$. In this case, the knowledge of an upper bound for p supports the elimination process [Stu98].

For our computation examples, we shall focus on the most general setting $q = 0$. It will turn out that the simplification strategies described above are so powerful, that in the parameter-free case we obtain a straightforward description of the class of valued fields in which the corresponding formula holds.

6 Example computations

All our computations have been carried out on a Sun Ultra-1 Model 140 using 64 MB of memory. The smallest measurable CPU time is 10 ms.

6.1 P-adic balls

With respect to p-adic absolute values, we consider open balls

$$\{\, x \in \mathbb{Q} : |x - a|_p < r \,\}$$

of radius r. Due to the ultrametric triangle inequality, they have the following property: For each pair of given balls there is either one of them included in

[1] `http://www.fmi.uni-passau.de/~redlog/`

the other one, or they are disjoint. This is stated by the following formula:

$$\forall r_1 \forall r_2 \forall a \forall b (\exists x (r_1 \parallel x - a \wedge r_2 \parallel x - b \wedge r_1 \mid r_2) \longrightarrow$$

$$\forall y (r_2 \parallel y - b \longrightarrow r_1 \parallel y - a)).$$

For automatically proving the theorem uniformly for all p-adic valuations, we eliminate all the variables. Recall that we have to expect the result to contain constraints involving the constant p. In addition, there will be variable-free constraints such as $2 \sim 3$, which cannot necessarily be evaluated to truth values. So following the quantifier elimination, simplification is a substantial step for obtaining a comprehensible result here. In fact, quantifier-elimination without simplification yields after 869 s a result containing 287655 constraints. Automatic application of our simplification strategies during the elimination process decreases the computation time to 2.2 s and shrinks the result to 24 constraints. By inspection they can easily verified to be "true" for any valuation.

Fixing the valuation to v_2 or v_{100003} and using our simplification with the elimination we obtain "true" after 860 ms and 1100 ms, respectively.

6.2 Size of the residue field

We are going to construct a first-order formula over our language that holds if and only if $|K_v| = |R_v/I_v| \geq 5$, i.e., the residue field wrt. the valuation contains at least five elements. Since for p-adic valuations we have $K_v = \mathbb{Z}/p$, the result of our uniform elimination should then somehow state $p \geq 5$. It will be interesting to observe how this is encoded in valuation constraints.

Our approach is to state that $0, \ldots, 3$ represent separate residue classes, and furthermore claim the existence of some x which does not belong to any of these classes. For any residue field K_v we certainly have $\overline{0} \neq \overline{1}$, and it follows that $\overline{1} \neq \overline{2}$ and $\overline{2} \neq \overline{3}$. Similarly, $\overline{1} \neq \overline{3}$ will follow from $\overline{0} \neq \overline{2}$. It thus suffices to explicitly state $\overline{0} \neq \overline{2}$ and $\overline{0} \neq \overline{3}$. That is, $2 - 0$, $3 - 0 \notin I_v$, in other words $v(2) = 0 = v(1)$, $v(3) = 0 = v(1)$, which is in our language $2 \sim 1 \wedge 3 \sim 1$. In the same way, we describe $\overline{x} \neq \overline{0}, \ldots, \overline{x} \neq \overline{3}$ by $x - 0 \sim 1 \wedge \cdots \wedge x - 3 \sim 1$. Together, we obtain as input formula

$$\exists x (2 \sim 1 \wedge 3 \sim 1 \wedge x \sim 1 \wedge \cdots \wedge x - 3 \sim 1).$$

Without simplification the elimination of $\exists x$ yields 295 constraints within 50 ms. Using our simplification we obtain after 120 ms the following result consisting of only 23 constraints:

$$(3 \sim 1 \wedge 2 \sim 1) \vee$$
$$(7 \sim 1 \wedge 6 \sim 1 \wedge 5 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1) \vee$$
$$(5 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1) \vee$$
$$(11 \sim 1 \wedge 10 \sim 1 \wedge 6 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1) \vee$$
$$(7 \sim 1 \wedge 6 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1) \vee$$
$$(6 \sim 1 \wedge 5 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1).$$

Since this result happens to be in disjunctive normal form, we can automatically check for subsumption between the conjunctions. This yields after 10 ms the simple solution

$$3 \sim 1 \wedge 2 \sim 1,$$

stating in a nice way that our input formula holds for any p-adic valuation except for v_2 and v_3.

6.3 Properties of affine linear functions

So far we have only considered parameter-free decision problems. Our first parametric problem is concerned with the question which affine linear functions have a zero of a value between 1 and 1000:

$$\exists x(ax + b = 0 \wedge p \mid x \wedge x \mid p^{1000}).$$

This yields after 10 ms the following equivalent condition in the parameters:

$$ap^{999} + b = 0 \vee ap + b = 0 \vee (ap \mid b \wedge a \neq 0 \wedge b \mid ap^{1000}).$$

Extended quantifier elimination yields these conditions together with satisfying sample points:

$$\left[\begin{array}{ll} ap + b = 0 & x = p \\ ap^{999} + b = 0 & x = p^{999} \\ ap \mid b \wedge a \neq 0 \wedge b \mid ap^{1000} & x = -b/a \end{array}\right].$$

This computation also takes only 10 ms.

Our next example asks for a characterization for two affine linear functions the zeroes of which have the same value:

$$\exists x_1 \exists x_2 (a_1 x_1 + b_1 = 0 \wedge a_2 x_2 + b_2 = 0 \wedge x_1 \sim x_2).$$

This yields after 20 ms the following equivalent condition in the parameters a_1, b_1, a_2, and b_2:

$$
\begin{gathered}
(a_1 b_2 - a_2 b_1 = 0 \land a_2 \neq 0) \lor \\
(a_1 b_2 - a_2 b_1 = 0 \land a_1 \neq 0) \lor \\
(b_1 = 0 \land b_2 = 0 \lor a_1 + b_1 = 0 \land a_2 + b_2 = 0) \lor \\
(a_1 \sim b_1 \land a_1 \neq 0 \land a_2 + b_2 = 0) \lor \\
(a_1 b_2 + a_2 b_1 = 0 \land a_2 \neq 0) \lor \\
(a_1 b_2 + a_2 b_1 p = 0 \land a_2 \neq 0 \land b_2 = 0) \lor \\
(a_2 \neq 0 \land b_1 = 0 \land b_2 = 0) \lor \\
(a_1 + b_1 = 0 \land a_2 \sim b_2 \land a_2 \neq 0) \lor \\
(a_1 b_2 \sim a_2 b_1 \land a_1 \neq 0 \land a_2 \neq 0).
\end{gathered}
$$

Obtaining sample solutions is also no problem here.

6.4 Linear congruences over the integers

We consider the following system of linear congruences, which has been randomly generated:

$$
\begin{aligned}
33x_3 + 62 &\equiv 0 \pmod{p^{10}}, \\
13x_1 + 17x_3 + 10x_4 + 25x_5 + 51 &\equiv 0 \pmod{p^8}, \\
19x_2 + 54x_5 + 89 &\equiv 0 \pmod{p^7}, \\
88x_2 + 56x_5 + 74 &\equiv 0 \pmod{p^5}, \\
96x_1 + 94x_2 + 92x_3 + 50x_5 + 48 &\equiv 0 \pmod{p^2}.
\end{aligned}
$$

Notice that the prime p is not fixed. Our goal is to solve the system for concrete p over the integers $\mathbb{Z}$.

The system is feasible over $\mathbb{Z}$ if and only if it is feasible over the p-adic integers R_{v_p}. The latter feasibility can be checked by applying our quantifier elimination to the following formula:

$$
\begin{gathered}
\exists x_1 \exists x_2 \exists x_3 \exists x_4 \exists x_5 (1 \mid x_1 \land 1 \mid x_2 \land 1 \mid x_3 \land 1 \mid x_4 \land 1 \mid x_5 \land \\
p^{10} \mid 33x_3 + 62 \land p^8 \mid 13x_1 + 17x_3 + 10x_4 + 25x_5 + 51 \land \\
p^7 \mid 19x_2 + 54x_5 + 89 \land p^5 \mid 88x_2 + 56x_5 + 74 \land \\
p^2 \mid 96x_1 + 94x_2 + 92x_3 + 50x_5 + 48).
\end{gathered}
$$

Quantifier elimination with successive automatic application of Lemma 9 yields after 1.4 s the following necessary and sufficient condition for the feasibility of the system:

$$461 \sim 1 \wedge 11 \sim 1 \wedge 5 \sim 1 \wedge 3 \sim 1 \wedge 2 \sim 1,$$

that is $p \notin \{461, 11, 5, 3, 2\}$. Extended quantifier elimination yields in addition a sample solution in R_{v_p}. As usual, the timings for extended and non-extended quantifier elimination are the same:

$$x_1 = \frac{5683171}{2920896}, \quad x_2 = \frac{247}{922}, \quad x_3 = -\frac{62}{33},$$

$$x_4 = -\frac{2320471}{29208960}, \quad x_5 = -\frac{3213}{1844}.$$

Given a concrete prime p, we can now easily lift this solution as follows: Fix, for example, $p = 13$, recall that the greatest power of p in the input system is p^{10}, and consider x_1. We compute by the extended Euclidean algorithm (10 ms)

$$1 = 53840023598 \cdot 2920896 - 1140743 \cdot 13^{10}.$$

That is $53840023598 \cdot 2920896 \equiv 1 \pmod{I_{13^{10}}}$, hence

$$\begin{aligned} x_1 &\equiv \frac{5683171 \cdot 53840023598 \cdot 2920896}{2920896} \pmod{I_{13^{10}}} \\ &= 305982060751469258 \in \mathbb{Z}. \end{aligned}$$

The congruence modulo $I_{13^{10}}$ certainly implies the congruence modulo I_{13^k} for all $1 \leq k \leq 10$. This way, we automatically lift all solutions within 10 ms:

$$x_1 = 305982060751469258, \quad x_2 = 8457364288997,$$

$$x_3 = 259006863472, \quad x_4 = 51485905148020710,$$

$$x_5 = 166462558935687.$$

Notice that the essential part of the computational work, namely the extended quantifier elimination, has been done uniformly for all p. In our example this uniform part makes up 99.93 percent of the computation time.

In general, one will obtain not only one but several conditions each associated with a sample point. Notice that after selecting a prime p, we can evaluate with our simplifier each of the conditions to either "true" or "false" and among the "true" cases pick a suitable p-adic integer solution.

7 Conclusions

We have discussed simplification strategies and corresponding implementation techniques for formulas over $\mathbb{Q}$ or $\mathbb{Q}_p$ with p-adic valuations. Our simplifier is implemented within the REDUCE package REDLOG. It is closely connected to the quantifier elimination for linear formulas there. Both procedures together enable us to parametrically check p-adic constraints for feasibility. The extended quantifier elimination also available in REDLOG together with the simplifier yields satisfying parametric sample points for feasible constraints. Optionally, all these computations can be performed uniformly for all p-adic valuations. An interesting application of our method is the solution of congruences modulo prime powers over the integers. Up to a straightforward final lifting step, this is done uniformly for all primes.

Acknowledgment

We are indebted to Volker Weispfenning for the idea of considering integer congruences within our framework, and for valuable discussions on lifting the solution to the integers.

References

[AK66] James Ax and Simon Kochen. Diophantine problems over local fields. *Annals of Mathematics*, 83:437–456, 1966. Part III.

[Coh69] Paul J. Cohen. Decision procedures for real and p-adic fields. *Communications in Pure and Applied Logic*, 25:213–231, 1969.

[DGS98] Andreas Dolzmann, Oliver Gloor, and Thomas Sturm. Approaches to parallel quantifier elimination. In Oliver Gloor, editor, *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation (ISSAC 98)*, pages 88–95, Rostock, Germany, Aug 1998. ACM, ACM Press, New York.

[DS97a] Andreas Dolzmann and Thomas Sturm. Guarded expressions in practice. In Wolfgang W. Küchlin, editor, *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation (ISSAC 97)*, pages 376–383, Maui, HI, July 1997. ACM, ACM Press, New York.

[DS97b] Andreas Dolzmann and Thomas Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[DS97c] Andreas Dolzmann and Thomas Sturm. Simplification of quantifier-free formulae over ordered fields. *Journal of Symbolic Computation*, 24(2):209–231, August 1997.

[DSW98] 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, Berlin, 1998.

[Dub92] Devdatt P. Dubhashi. Algorithmic investigations in p-adic fields. Technical Report 92-1301, Department of Computer Science, Cornell University, Ithaca, New York, September 1992. Ph.D Thesis.

[Ers65] Juri L. Ershov. On elementary theories of local fields. *Algebra i Logika Sem.*, 4(2):5–30, 1965.

[Mac76] Angus Macintyre. On definable subsets of p-adic fields. *Journal of Symbolic Logic*, 41(3):605–610, September 1976.

[Ost18] Alexander Ostrowski. Über einige Lösungen der Funktionalgleichung $\varphi(x) \cdot \varphi(y) = \varphi(xy)$. *Acta Mathematica*, 41:271–284, 1918.

[Stu98] Thomas Sturm. Linear problems in valued fields. Technical Report MIP-9715, FMI, Universität Passau, D-94030 Passau, Germany, November 1998.

[Wei84] Volker Weispfenning. Quantifier elimination and decision procedures for valued fields. In G. H. Müller and M. M. Richter, editors, *Models and sets (Aachen, 1983)*, volume 1103 of *Lecture Notes in Mathematics*, pages 419–472. Springer-Verlag, Berlin, Heidelberg, 1984.

[Wei88] Volker Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1–2):3–27, February–April 1988.