Computability and Differential Fields: a Tutorial

Russell Miller*

June 9, 2008

1 Introduction

Computability theory applies a rigorous definition of the notion of an algorithm to determine which mathematical functions can be computed and which cannot. The main concepts in this area date back to Alan Turing, who during the 1930's gave the definition of what is now called a *Turing machine*, along with its generalization, the *oracle Turing machine*. In the ensuing seventy years, mathematicians have developed a substantial body of knowledge about computability and the complexity of subsets of the natural numbers. It should be noted that for most of its history, this subject has been known as *recursion theory*; the terms *computable function* and *recursive function* are to be treated as interchangeable.

Computable model theory applies the notions of computability theory to arbitrary mathematical structures. Pure computability normally considers functions from $\mathbb{N}$ to $\mathbb{N}$, or equivalently, subsets of finite Cartesian products $\mathbb{N} \times \cdots \times \mathbb{N}$. Model theory is the branch of logic in which we consider a structure (i.e. a domain of elements, with appropriate functions and relations on that domain) and examine how exactly the structure can be described in our language, using symbols for those functions and relations, along with the usual logical symbols such as negation, conjunction, $(\exists x)$, and $(\forall x)$. To fit this into the context of computability, we usually assume that the domain

*The author was partially supported by PSC-CUNY grants numbered PSCREG-38-967, 68470-00-37, and 80209-04-12.

is $\mathbb{N}$, so that the functions and relations become the sort of objects usually studied by computability theorists.

In this paper, based on a tutorial at the Second International Workshop on Differential Algebra and Related Topics on April 12, 2007, we mostly consider the specific case of a computable field. Our goal is to encourage the study of computable differential fields, about which relatively little is known; the use of ordinary fields as examples allows us to demonstrate in a concrete way how the techniques of computability theory may be applied to such structures and what sort of results can be derived. Since We include a selection of results about computable fields, with proofs included or sketched, to help the reader imagine what corresponding results might or might not hold for computable differential fields. These results should be assumed to be folklore unless specific attribution is made. In particular, apart from the work summarized in Section 6, the author makes no claim to originality.

A related article by the same author appears as [10]. It contains a further selection of results about computable fields, mostly disjoint from those given here, along with a introductory catalogue of standard results from pure computability theory, mostly without proofs, and a partial analysis, with examples, of which fields actually are computable. Since it is readily available, we have elected not to reproduce the same introduction here; rather we refer the reader to [10] for it. We have made sure to use the same terminology and notation here that is used there. Rigorous introductions to computability theory can be found in any standard text on computability, including [8], [15], and [16].

Many thanks are due to Professors Phyllis Cassidy, Richard Churchill, Li Guo, William Keigher, Jerry Kovacic, and William Sit, the organizers of the the Second International Workshop on Differential Algebra and Related Topics and editors of the conference proceedings volume.

2 Preliminaries

For us the natural numbers include 0; for readability we use the standard symbol $\mathbb{N}$ rather than the logician's symbol ω for the set of natural numbers. A function $f : \mathbb{N} \to \mathbb{N}$ is said to be *computable* if there is a Turing machine which computes f. Specifically, on each input $n \in \mathbb{N}$, the program should eventually halt with output $f(n)$. More generally, we consider *partial functions* $\varphi : \mathbb{N} \to \mathbb{N}$, for which (despite the similarity of notation) the domain is

allowed to be any subset of $\mathbb{N}$. We write $\varphi(n)\downarrow$, and say that $\varphi(n)$ *converges*, if $n \in \mathrm{dom}(\varphi)$; otherwise $\varphi(n)$ *diverges*, written $\varphi(n)\uparrow$.

A subset $S \subseteq \mathbb{N}$ is said to be *computable* iff its characteristic function χ_S is computable. S is *computably enumerable*, abbreviated *c.e.*, if it is empty or is the range of some total computable function f. Intuitively, this says that there is a mechanical way to list out the elements of S: just compute $f(0)$, then $f(1)$, etc., and write each one on our list.

Fact 2.1 *A set S is computably enumerable iff S is the range of a partial computable function, iff there is a computable set R such that $S = \{x : \exists y_1 \cdots \exists y_k \ (x, y_1, \ldots y_k) \in R\}$, iff S is the domain of a partial computable function.*

Fact 2.2 *There exists a c.e. set which is not computable.*

For the R in Fact 2.1, we need to consider subsets of Cartesian products $\mathbb{N}^k$ as well. For this we use the function

$$\beta_2(x, y) = \frac{1}{2} \cdot (x^2 + y^2 + x + 2xy + 3y),$$

which is a bijection from $\mathbb{N}^2$ onto $\mathbb{N}$. If π_1 and π_2 are projections, then both functions $\pi_i \circ \beta_2^{-1}$ are computable. We use β_2 to treat $\mathbb{N}^2$ as though it were just $\mathbb{N}$, and then define $\beta_3(x, y, z) = \beta_2(x, \beta_2(y, z))$ and so on. Indeed, the bijection β defined by

$$\beta(x_1, \ldots, x_k) = \beta_2(k, \beta_k(x_1, \ldots, x_k))$$

maps the set $\mathbb{N}^*$ of all finite sequences of natural numbers bijectively onto $\mathbb{N}$. This gives us a way of allowing polynomials from $\mathbb{N}[X]$ to be the inputs to a computable function.

We have a computable list of all partial computable functions, which we usually write as $\varphi_0, \varphi_1, \ldots$, with $\varphi_e(x) = \varphi(e, x)$ for a fixed universal partial computable function φ. Likewise, using Fact 2.1, this yields a computable list of all computably enumerable sets $W_0, W_1, \ldots$, with $W_e = \mathrm{dom}(\varphi_e)$.

When considering fields, we normally work in a language which includes the addition and multiplication symbols, regarded as binary functions, and two constant symbols to name the identity elements, along with all standard logical symbols. A field $\mathcal{F}$ then consists of a set F of elements (called the *domain* of the field, but not to be confused with the separate notion of a

ring without zero-divisors), with two elements of F distinguished as the two constants, and with two binary functions on F, represented by the symbols $+$ and $\cdot$, all satisfying the standard field axioms.

For a field to be computable, we want to be able to compute the two field operations in this language. Our notion of computability is defined over $\mathbb{N}$, so we index the elements of the field using $\mathbb{N}$. Of course, this immediately eliminates all uncountable fields from the discussion! Section 6 mentions a possible approach to this problem.

Definition 2.3 A *computable field* is a field $\mathcal{F}$ with domain $\{a_0, a_1, \ldots\}$ and with two computable total functions f and g such that for all $i, j \in \mathbb{N}$,

$$a_i + a_j = a_{f(i,j)} \quad \text{and} \quad a_i \cdot a_j = a_{g(i,j)}.$$

In most of computable model theory one takes $\mathbb{N}$ itself to be the domain. However, we will wish to use the symbols 0 and 1 to refer to the identity elements of the field (and perhaps 2 to refer to the sum $1 + 1$, etc.), so we use the notation a_i to avoid confusion.

In the language of differential fields, we have one or several additional unary function symbols in the language, representing the differential operators, and additional axioms which they must satisfy. A *computable differential field* therefore should be a computable field in which each differential operator δ is likewise given by a computable total function, say h, such that $\delta(a_i) = a_{h(i)}$ for all $i \in \mathbb{N}$. If there were infinitely many differential operators, then we would require some sort of uniformity in giving the programs which compute them all, but in this paper we will stick to finite languages. Mostly we will be discussing pure fields, since far more is known about algorithmic question for them; indeed, one of our goals is to inspire further research on computable differential fields, as we discuss in Section 5.

Notice that it is quite possible for a computable field to be isomorphic to a field that is not computable. So we should not speak of (the isomorphism type of) a field as being computable; rather we say that a field (or its isomorphism type) is *computably presentable* if it is isomorphic to a computable field.

A standard question for a field $\mathcal{F}$ is the existence of a *splitting algorithm* for $\mathcal{F}$. A computable field $\mathcal{F}$ has a splitting algorithm iff the set of irreducible polynomials in $\mathcal{F}[X]$ is a computable set. (Again, we use β as a canonical translation between $\mathbb{N}^*$, i.e. the set of polynomials, and $\mathbb{N}$.)

Kronecker gave a splitting algorithm for $\mathbb{Q}$ itself. In fact, he showed that every finite extension of $\mathbb{Q}$ has a splitting algorithm, using the following.

Theorem 2.4 (Kronecker [7]; see also [2]) *If a field L has a splitting algorithm, then so does $L(X)$ for any X transcendental over L. When x is algebraic over L, again $L(x)$ has a splitting algorithm, but it requires knowledge of the minimal polynomial of x over L.*

In [10] we define fields $\mathcal{E}_K = \mathbb{Q}[\sqrt{p_n} \ : \ n \in K]$ and $\mathcal{E}_{\overline{K}} = \mathbb{Q}[\sqrt{p_n} \ : \ n \notin K]$, using the c.e. set K from Fact 2.2, and show that $\mathcal{E}_K$ is computably presentable, but that the presentation has no splitting algorithm, while $\mathcal{E}_{\overline{K}}$ is not computably presentable. This yields the fact below.

Fact 2.5 *There exists a computable field without a splitting algorithm.*

We again encourage the reader to consult the early sections of [10], where most results in this section are expanded and better explained.

3 Transcendence Bases

Every field $\mathcal{F}$ has a prime (i.e. smallest) subfield $\mathcal{F}_0$, either $\mathbb{Z}_p$ for fields of characteristic p, or $\mathbb{Q}$ when the characteristic is 0. The subfield $\mathcal{F}_0$ always forms a computably enumerable subset of $\mathcal{F}$, and indeed in positive characteristic it is finite, hence computable. It is natural to ask which field elements are algebraic over the prime subfield, and to hope for a decision procedure for this question.

If there exists a computable transcendence basis B for a computable field $\mathcal{F}$, then we will have a decision procedure for the set of elements algebraic over $\mathcal{F}_0$. In particular, writing $B = \{X_i : i \in I\}$ for an appropriate initial segment I of $\mathbb{N}$, every $x \in \mathcal{F}$ will be algebraic over the purely transcendental extension $\mathcal{F}_0(X_0, X_1, \ldots)$. So, given an $x \in \mathcal{F}$, we search for an n and a finite tuple $c_0, \ldots, c_d$ of elements of $F_0(X_0, \ldots, X_n)$ such that $c_d x^d + \cdots + c_0 = 0$ in $\mathcal{F}$. Eventually we must find such a tuple. Then we apply the splitting algorithm from Kronecker's Theorem 2.4 for $\mathcal{F}_0(X_0, \ldots, X_n)$, and if this polynomial is reducible, we check which factor has x as a root and repeat the process with that factor. Eventually we find the minimal polynomial $p(X)$ of x over $\mathcal{F}_0(X_0, \ldots)$. Now x is algebraic iff $p(X) \in \mathcal{F}_0[X]$. For each coefficient c in $p(X)$, we may find an expression of c as a quotient of polynomials in $\mathcal{F}_0[X_0, \ldots, X_m]$ for some m. With our splitting algorithm, we may factor these until the quotient is in lowest terms (up to constants in $\mathcal{F}_0$). Then $c \in \mathcal{F}_0$ iff the numerator and denominator of this quotient both have degree

0 as polynomials. If this holds for every coefficient in $p(X)$, then x is algebraic over $\mathcal{F}_0$; otherwise not.

We point out that in the foregoing procedure, it is sufficient for B to be computably enumerable, rather than actually computable. Also, we could do the same procedure not only for the prime subfield $\mathcal{F}_0$, but for any computably enumerable subfield over which we have a splitting algorithm. So the question of algebraicity over a (nice) subfield hinges mainly on the existence of a computable transcendence basis for the field over the subfield. Indeed, the converse is clear: if we can decide algebraicity of arbitrary elements over $\mathcal{F}_0$, then we can compute a transcendence basis B, just by going through all elements $a_0, a_1, \ldots$ of $\mathcal{F}$ and checking whether each a_n is algebraic over the extension of $\mathcal{F}_0(X_0, \ldots, X_k)$, where the X_i are those elements a_m with $m < n$ which have already been placed in B. If a_n is transcendental over this extension, we add it to B; otherwise not. This works not just for the prime subfield, but for any c.e. subfield $\mathcal{F}_0$ with a splitting algorithm.

So we ask next whether every computable field $\mathcal{F}$ has a computable transcendence basis over its prime subfield $\mathcal{F}_0$, or at least a computably enumerable one. Half an answer is found fairly quickly.

Lemma 3.1 *Every computable field $\mathcal{F}$ has a co-c.e. transcendence basis, i.e. a transcendence basis whose complement (in the domain of $\mathcal{F}$) is computably enumerable.*

Proof. A co-c.e. set S is built by elimination: one starts with the assumption that $S = \mathbb{N}$, and then runs a computable process under which certain elements drop out of S (and thus enter $\overline{S}$, which is c.e.). In this situation, we let $\overline{B}$ be the set of all those $a_n \in \mathcal{F}$ such that there exist coefficients in $\mathcal{F}_0$ for a polynomial $p(X_0, \ldots, X_n) \in \mathcal{F}_0[X_0, \ldots, X_n]$ such that:

- $p(a_0, \ldots, a_n) = 0$; and
- not all coefficients of $p(a_0, \ldots, a_{n-1}, X_n)$ equal zero.

This condition is defined by an existential formula (since $\mathcal{F}_0$ itself is always c.e.), and so $\overline{B}$ is c.e. and B itself is co-c.e. Moreover, B is algebraically independent over $\mathcal{F}_0$, since every $a_n \in B$ is transcendental over $\mathcal{F}_0(a_0, \ldots, a_{n-1})$, and must be a transcendence basis, since every $a_n \notin B$ is algebraic over $\mathcal{F}_0(a_0, \ldots, a_{n-1})$, hence (by induction) over some subset of B. ∎

In order to build a computable field with no computable transcendence basis, therefore, we focus on making sure that no transcendence basis is c.e. Actually, since there are uncountably many transcendence bases (assuming the transcendence degree is infinite) and only countably many c.e. sets, it will be easier to ensure that no c.e. set is a transcendence basis.

Theorem 3.2 (Metakides-Nerode) *There exists a computable field with no computably enumerable transcendence basis.*

Proof. A full proof of this result was given by Metakides and Nerode in [9]. Here we sketch the proof. Recall that we can enumerate all c.e. sets simultaneously as $W_0, W_1, \ldots$. Our goal is to ensure, for each e, that W_e will not be a transcendence basis for $\mathcal{F}$ over its prime field $\mathcal{F}_0$. We will make $\mathcal{F}$ have infinite transcendence degree over $\mathcal{F}_0$, since otherwise it would certainly have a computable transcendence basis, and so we need not worry about any finite set W_e.

If W_e is infinite, our strategy is the following. We start building a computable copy of the purely transcendental extension $\mathbb{Q}(X_0, X_1, \ldots)$ of $\mathbb{Q}$, but slowly – that is, after finitely many steps, we should only have defined finitely many values in the (infinite) addition and multiplication tables for $\mathcal{F}$. Simultaneously, we enumerate all sets W_e. If a particular W_e is to be a transcendence basis, it must eventually enumerate some element not in the subfield $\mathcal{F}_e = \mathbb{Q}(X_0, \ldots, X_{2e})$. We will watch for this element, and since we have a splitting algorithm for $\mathcal{F}_e$, we will recognize it when it appears. Call it y_e. (If no such y_e ever appears, then either W_e was finite, or W_e contained more than $2e$ elements of $\mathcal{F}_e$. Each of these would ensure that W_e is not a transcendence basis, so we need not do anything unless such a y_e appears.) When and if we find such a y_e, we wish to make it algebraic over F_0, thereby voiding W_e's hope of being algebraically independent. To do so, we go through the infinitely many elements of $\mathbb{Q}$ until we find some $q \in \mathbb{Q}$ such that we can map y_e to q without disrupting any of the finitely many additions and multiplcations already defined in $\mathcal{F}$. For instance, if a_8 is already defined to be 4, i.e. $1 + 1 + 1 + 1 = a_8$ is already in the addition table, then we should not map y_e to 4, lest the distinct elements y_e and a_8 wind up equal in $\mathcal{F}$. If we have also already defined $y_e \cdot y_e = a_{20}$, then y_e should not be mapped to the negative of $(1 + 1)$, for then a_{20} and a_8 would have to be equal in $\mathcal{F}$. Fortunately, with only finitely many additions and multiplications defined as yet, and with infinitely many $q \in \mathbb{Q}$ to choose from, we will eventually find

some q such that we can define the addition to make y_e represent this q in $\mathcal{F}_0$ without changing any of the existing additions or multiplications. Indeed, a sufficiently large integer would suffice.

Intuitively, we simply tell W_e: "Ha! You thought that this y_e was transcendental. Well, it's really the integer 10^{17}." Of course, had W_e waited longer to enumerate y_e, our addition and multiplication functions might have precluded the possibility of y_e being 10^{17}, but at any finite stage of the process, we will be able to find an integer q sufficiently large that our operations as so far defined do not give any reason why y_e could not equal q in $\mathcal{F}$. So in fact, in the field $\mathcal{F}$, every infinite c.e. set will intersect the "naturals," i.e. the set $\{(1 + \cdots + 1) \quad (n \text{ times}) : n \in \mathbb{N}\}$. Thus we certainly cannot have any infinite c.e. transcendence basis.

A few points here should be cleared up. First, when we switch and make some y_e algebraic, it changes the splitting algorithm of $\mathcal{F}_i$ for every $i > e$. Of course, if we had already put y_i into $\mathcal{F}_0$, this change does not make y_i transcendental again, so we do not undo any previous accomplishments. (If it did, a method called a *finite injury priority* construction would be needed here.) Moreover, having made the change, we do know the minimal polynomial of y_e over $\mathcal{F}_0$, so we still know all the splitting algorithms we will need in the future; we just remember which X_n are still transcendental and which are not.

Second, we claimed that $\mathcal{F}$ has infinite transcendence degree over $\mathcal{F}_0$. The reason for this is that the elements $X_0, \ldots, X_{2e}$ originally were algebraically independent over $\mathcal{F}_0$, with transcendence degree $(2e + 1)$, and only the sets $W_0, \ldots, W_{e-1}$ can have changed that. Since each of the changes reduces the transcendence degree by 1, we see that $\{X_0, \ldots, X_{2e}\}$ still has transcendence degree at least $(e + 1)$ over $\mathcal{F}_0$. Since this holds for every e, the entire field $\mathcal{F}$ has infinite transcendence degree. This completes our sketch of the proof of Theorem 3.2. ∎

4 Algebraic Closures and Rabin's Theorem

Every countable field has countable algebraic closure, i.e. embeds into a countable algebraically closed field which is algebraic over the image of the embedding. If the field is computable, we would like to claim that its algebraic closure must also be computable.

Stated naively, the question is easy to answer. A countable algebraically

closed field is characterized by its transcendence degree and its characteristic. Using splitting algorithms, it is not difficult to build computable fields isomorphic to the algebraic closures of $\mathbb{Q}$, of $\mathbb{Q}(X_0, \ldots, X_n)$, and of $\mathbb{Q}(X_0, \ldots)$, and the same holds with any $\mathbb{Z}_p$ in place of $\mathbb{Q}$. So every countable algebraically closed field is computably presentable, and therefore the answer to our original question is clearly affirmative.

In fact, this makes it clear that *every* countable field has a computable algebraic closure, even fields, such as the $\mathcal{E}_{\overline{K}}$ from Section 2, which have no computable presentation at all. This feels wrong to us, and suggests that we should revise our question. Not only do we want the algebraic closure $\overline{\mathcal{F}}$ to have a computable presentation, but we also want the original field $\mathcal{F}$ to sit inside $\overline{\mathcal{F}}$ in a nice way. Ideally, we would like $\mathcal{F}$ to be a computable subfield of $\overline{\mathcal{F}}$. If this fails, a computably enumerable subfield would be the next best thing. For $\mathcal{E}_{\overline{K}}$, even that is impossible: the reader is encouraged to prove the following lemma for herself.

Lemma 4.1 *Every computably enumerable subfield of a computable field $\mathcal{F}$ is itself computably presentable, indeed in such a way that there is a computable isomorphism from the computable presentation onto the subfield of $\mathcal{F}$.*

The definitive answer to the real question about computable algebraic closures was given in 1960 by Michael Rabin (see [13]). We give his name to the type of embedding we wish to consider.

Definition 4.2 Let $\mathcal{F}$ and $\mathcal{E}$ be computable fields. A function $g : \mathcal{F} \to \mathcal{E}$ is a *Rabin embedding* if:

- g is a homomorphism of fields; and
- $\mathcal{E}$ is both algebraically closed and algebraic over the image of g; and
- g is a computable function. (More precisely, there is a total computable h with $g(a_n) = b_{h(n)}$ for all n, where $\mathcal{F}$ has domain $\{a_0, a_1, \ldots\}$ and $\mathcal{E}$ has domain $\{b_0, b_1, \ldots\}$.)

Thus $\mathcal{E}$ is the algebraic closure of $\mathcal{F}$ in a strong way: we actually have $\mathcal{F}$ as a subfield of $\mathcal{E}$, using the computable isomorphism g, and that subfield is computably enumerable, since (the indices of) its elements form the range of a total computable function. If we cannot compute a transcendence basis for

$\mathcal{F}$, as for the field built in Theorem 3.2, then it is not obvious that a Rabin embedding of $\mathcal{F}$ exists. Indeed, if we are simply given a computable presentation of the algebraic closure of $\mathcal{F}$, there may be no computable embedding of $\mathcal{F}$ into that closure. However, Rabin proved that a Rabin embedding must exist, essentially by building a computable presentation of it around the computable field $\mathcal{F}$.

Theorem 4.3 (Rabin [13]) *Let $\mathcal{F}$ be any computable field.*

1. *There exists a computable ACF $\overline{\mathcal{F}}$ with a Rabin embedding of $\mathcal{F}$ into $\overline{\mathcal{F}}$.*

2. *For every Rabin embedding g of $\mathcal{F}$ (into any computable ACF $\mathcal{E}$), the image of g is a computable subset of $\mathcal{E}$ iff $\mathcal{F}$ has a splitting algorithm.*

Proof. The heart of the proof of this theorem is the construction of the $\overline{\mathcal{F}}$ and g required by Part (1). This is extremely involved, and we refer the reader to [13], where this result appears as Theorem 7. The portion of the theorem usually quoted is Part (2), which has a very nice and readily understandable proof.

($\Rightarrow$): Assume that the image $g(\mathcal{F})$ is computable within $\overline{\mathcal{F}}$, and fix any polynomial $p(X) \in \mathcal{F}[X]$. Since g is computable, we can find the polynomial $q(X) \in \overline{\mathcal{F}}[X]$ whose coefficients are the images under g of the coefficients of $p(X)$. Since $\overline{\mathcal{F}}$ is algebraically closed, we search until we find all roots $r_1, \ldots, r_d$ of $q(X)$ in $\overline{\mathcal{F}}$, with repetitions allowed, i.e. with $d = \deg(q)$. (Roots of multiplicity > 1 are not a hindrance: when we find a root, we factor it out of $q(X)$ and continue to search for roots of the remaining polynomial.) Thus

$$q(X) = (X - r_1) \cdots (X - r_d).$$

Now we consider all $(2^d - 2)$ nontrivial subsets $S \subset \{1, \ldots, d\}$. If any polynomial

$$\Pi_{i \in S}(X - r_i)$$

has all its coefficients in the image of g, then we have a factorization of $q(X)$ within the polynomial ring $(g(\mathcal{F}))[X]$, and so $p(X)$ was reducible in $\mathcal{F}[X]$. Conversely, if there is no such S, then $p(X)$ was irreducible. Thus the computability of the image of g yields a splitting algorithm for $\mathcal{F}$.

($\Leftarrow$): Now suppose we have a splitting algorithm for $\mathcal{F}$, and fix any $x \in \mathcal{E}$. We search for a polynomial $p(X) \in \mathcal{F}[X]$ such that its image $q(X) \in \mathcal{E}[X]$

under g (just as before) satisfies $q(x) = 0$. Since $\mathcal{E}$ is algebraic over the image of g, we must eventually find such a $p(X)$. Then use the splitting algorithm for $\mathcal{F}$ to factor $p(X)$ into factors $p_1(X), \ldots, p_k(X)$ irreducible in $\mathcal{F}[X]$, with images $q_1(X), \ldots, q_k(X)$ in $\mathcal{E}[X]$ under g. Now x must be a root of one (and only one) $q_i(X)$, and x lies in the image of g iff that $q_i(X)$ is linear. ∎

A surprising corollary, which we believe appears for the first time in [11], is the following. Rabin's Theorem, while not actually required for this result, simplifies the proof.

Corollary 4.4 *Let $\mathcal{E}$ and $\mathcal{F}$ be isomorphic computable fields, each algebraic over its prime field. Then $\mathcal{E}$ has a splitting algorithm iff $\mathcal{F}$ does.*

So for algebraic fields, the existence of a splitting algorithm depends only on the isomorphism type, not on the specific computable presentation chosen.

Proof. Fix Rabin embeddings g and h of $\mathcal{E}$ and $\mathcal{F}$ into their computable algebraic closures $\overline{\mathcal{E}}$ and $\overline{\mathcal{F}}$. Suppose that $\mathcal{E}$ has a splitting algorithm, so that Theorem 4.3 shows $g(\mathcal{E})$ to be computable within $\overline{\mathcal{E}}$, and fix any $x \in \overline{\mathcal{F}}$. Start enumerating the prime field $\mathcal{F}_0$ of $\mathcal{F}$, and search for any polynomial $p(X) \in \mathcal{F}_0[X]$ such that the image $q(X)$ of p under h satisfies $q(x) = 0$. Since $\overline{\mathcal{F}}$ is algebraic over $g(\mathcal{F})$ and $\mathcal{F}$ is algebraic over $\overline{\mathcal{F}}_0$, such a $p(X)$ must exist. Now the unique isomorphism f from $\mathcal{F}_0$ onto the prime field $\mathcal{E}_0$ of $\mathcal{E}$ is computable, so we may find the image $r(X) \in \mathcal{E}_0[X]$ of p under f. Since $\mathcal{E}$ and $\mathcal{F}$ are isomorphic, $p(X)$ has exactly as many roots in $\mathcal{F}$ as $r(X)$ has in $\mathcal{E}$. Find all $\deg(r)$-many roots in $\overline{\mathcal{E}}$ of the image of $r(X)$ under g, and check how many of them lie in $g(\mathcal{E})$, using the computability of $g(\mathcal{E})$. Then enumerate $h(\mathcal{F})$ until the same number of roots of $q(X)$ have appeared in $h(\mathcal{F})$. Now $x \in h(\mathcal{F})$ iff x is one of those roots. Thus $\mathcal{F}$ has a splitting algorithm, by Theorem 4.3. ∎

5 Computable Differential Algebra

As mentioned in Section 2, a differential field is *computable* if the underlying field is computable and all differential operators are defined by computable unary functions. Of course, just as any field becomes a differential field under the definition $\delta x = 0$, any computable field becomes a computable differential field under that same definition. Moving past this trivial case, we note

for every n, there is a computable presentation of the rational function field $\mathbb{Q}(X_0, \ldots, X_n)$ under the partial differential operators $\frac{\partial}{\partial X_i}$. The same holds for the field of algebraic functions in the same indeterminates; these comments simply reflect that differentiation of such functions can be performed algorithmically.

If we extend to the field $\mathbb{Q}(X_0, X_1, \ldots)$ with a countable transcendence basis, and add all partial differential operators $\frac{\partial}{\partial X_i}$, the same result holds. One should note, however, that now we have an *infinite language*, with countably many function symbols, and so for a computable presentation, we must require not only that every individual function $\frac{\partial}{\partial X_i}$ be computable, but that they be so *uniformly*: there should exist a single total computable function h such that for all i and j, $\frac{\partial}{\partial X_i}(a_j) = a_{h(i,j)}$. When we have only finitely many i, of course, uniform computability is equivalent to the computability of every single operator.

And what of differentially closed fields? The author is grateful to Michael Singer for pointing out the following theorem.

Theorem 5.1 (Harrington [5]) *The differential closure of a computable differential field of characteristic* 0 *is itself computably presentable. So is the differential closure of a computable differentially perfect field of characteristic* p.

The proof of Harrington's Theorem also provides a computable embedding of the original field into the differential closure, such that the closure is differentially algebraic over the image of the embedding. Thus, this embedding is analogous to a Rabin embedding, and its image is c.e. in the domain of the computable differential closure. For differential fields there is at present no known analogue of part (2) of Rabin's Theorem.

Golubitsky and the author have conjectured that the Ritt problem might play a role for differential fields similar to that of the splitting algorithm for pure fields. The Ritt problem has several equivalent formulations, including the following.

Question 5.2 (Ritt Problem) *In a computable differential field* F*, does there exist an algorithm which, given a finite set of differential polynomials over* F*, determines whether the radical differential ideal generated by those polynomials is prime?*

If we restrict this problem to the case of a computable field with the trivial derivation, and to polynomials in a single variable, then from the finite set of

polynomials we can determine a single polynomial generating the same radical ideal, and primality of the radical ideal is then equivalent to irreducibility of the single generator (apart from the possibility that the generator f is itself a power of a smaller polynomial, which we can check by testing f and f' for common factors). So in this sense, the Ritt problem can be seen as a generalization of the question of existence of a splitting algorithm. Much more work remains if we hope to justify any claim that this is the "right" generalization, of course, but the argument above, along with Fact 2.5, does show that there is no general algorithm for solving the Ritt problem in all computable differential fields.

Another question is whether the algebraic closure of a computable differential field $\mathcal{F}$ forms a computable subfield of the computable differential closure.

Lemma 5.3 *Let $\mathcal{F}$ be a computable differential field in which all of the (finitely many) derivations are trivial. Then within the computable presentation of the differential closure $\mathcal{E}$ of $\mathcal{F}$ given in Theorem 5.1, the algebraic closure of $\mathcal{F}$ is a computable subfield, containing precisely the* constants of *$\mathcal{E}$, i.e. those elements $v \in \mathcal{E}$ such that $\delta v = 0$ for all derivations δ.*

Proof. For differential algebraists this is immediate, but we give a logician's proof. First suppose $v \in \mathcal{E}$ is in the algebraic closure of $\mathcal{F}$ (which we may view as a differential subfield of $\mathcal{E}$). Let $\sum_{i=0}^{n} c_i V^i$ be the minimal (algebraic) polynomial of v over $\mathcal{F}$. Applying any δ, we get $c_1\delta v + 2c_2 v\delta v + \cdots + nc_n v^{n-1}\delta v = 0$. Hence $q(v) \cdot \delta v = 0$, where $q(V)$ is an algebraic polynomial over $\mathcal{F}$ of degree $< n$. But then $q(v) \neq 0$, so $\delta v = 0$.

For the converse, we appeal to model theory. The set of formulas

$$S = \{\delta V = 0 : \text{all derivations } \delta\} \cup \{p(V) \neq 0 : \text{all nonconstant } p(V) \in \mathcal{F}[V]\}$$

forms an (incomplete) *nonprincipal type*. Each finite subset of S is satisfiable by some element of $\mathcal{E}$ which fails to satisfy the entire set. (That is, no finite subset of S implies all the formulas in S.) But the differential closure of $\mathcal{F}$ is by definition the prime model extension of $\mathcal{F}$, and as a prime model, it must realize only principal types. This precludes the possibility that any $v \in \mathcal{E}$ satisfies all the formulas of S, and so every $v \in \mathcal{E}$ not in the algebraic closure of $\mathcal{F}$ must have $\delta v \neq 0$ for some derivation δ.

For the model theory here, we suggest [1] and [6]. The model-theoretic definition of differential closure appears on p. 134 of [19]. ∎

For a computable differential field $\mathcal{F}$ in general, the algebraic closure will be computably enumerable within the differential closure, because the condition for x to lie in the algebraic closure is existential: one needs an algebraic polynomial with coefficients in the image of $\mathcal{F}$ for which x is a root. (The image of $\mathcal{F}$ in the differential closure is at least c.e, so we can search systematically for such a polynomial.) Of course, x always satisfies a differential polynomial over $\mathcal{F}$. An appropriate notion of splitting algorithm for differential fields, if one exists, ought to allow us to determine the minimal polynomial for x and check whether it is algebraic. For more on approaches to such problems, see the work of Ritt in [14].

6 Local Computability

The foregoing discussion only applied to computable fields. Other countable fields can be considered if one relativizes the notion of computability to allow an oracle, and the results from preceding sections would generally carry over to that case. However, computable model theory has always restricted itself to countable structures, essentially because the nature of Turing machines and computations in finite time allows only countably many inputs to such a machine. The author is enthusiastic about his own current project on *locally computable structures*, i.e. mathematical structures, quite possibly uncountable, whose finitely generated substructures are all computably presentable in a uniform way. Details are available in [12], but we offer a summary here.

Let $\mathcal{S}$ be any structure. (Feel free to think of $\mathcal{S}$ as a differential field.) A simple cover of $\mathcal{S}$ is simply a list of the finitely generated substructures of $\mathcal{S}$ (resp. differential subfields), up to isomorphism, with no attention paid to any relations between those substructures.

Definition 6.1 A *simple cover of* $\mathcal{S}$ is a (finite or countable) collection $\mathfrak{A}$ of finitely generated structures $\mathcal{A}_0, \mathcal{A}_1, \ldots$, such that:

- every finitely generated substructure of $\mathcal{S}$ is isomorphic to some $\mathcal{A}_i \in \mathfrak{A}$; and
- every $\mathcal{A}_i \in \mathfrak{A}$ embeds isomorphically into $\mathcal{S}$.

$\mathfrak{A}$ is *computable* if every $\mathcal{A}_i \in \mathfrak{A}$ is a computable structure whose domain is an initial segment of $\mathbb{N}$, and *uniformly computable* if the sequence $\langle (\mathcal{A}_i, \vec{a}_i) \rangle_{i \in \mathbb{N}}$,

where each $\vec{a}_i$ is a finite generating set for $\mathcal{A}_i$, can be given uniformly by a single computable function.

To consider interactions among the distinct finitely generated substructures of $\mathcal{S}$, we say that a map $f : \mathcal{A}_i \hookrightarrow \mathcal{A}_j$ *lifts* to an inclusion $\mathcal{B} \subseteq \mathcal{C}$ within $\mathcal{S}$, via β and γ, if $\beta : \mathcal{A}_i \to \mathcal{B}$ and $\gamma : \mathcal{A}_j \to \mathcal{C}$ are isomorphisms such that $\beta = \gamma \circ f$.

Definition 6.2 A *cover of* $\mathcal{S}$ consists of a simple cover $\mathfrak{A} = \{\mathcal{A}_0, \mathcal{A}_1, \ldots\}$ of $\mathcal{S}$, along with sets $I_{ij}^{\mathfrak{A}}$ (for all $\mathcal{A}_i, \mathcal{A}_j \in \mathfrak{A}$) of injective homomorphisms $f : \mathcal{A}_i \hookrightarrow \mathcal{A}_j$, such that:

- for all substructures $\mathcal{B} \subseteq \mathcal{C}$ of $\mathcal{S}$, there exist $i, j \in \mathbb{N}$ and $f \in I_{ij}^{\mathfrak{A}}$ and isomorphisms $\beta : \mathcal{A}_i \twoheadrightarrow \mathcal{B}$ and $\gamma : \mathcal{A}_j \twoheadrightarrow \mathcal{C}$ such that f lifts to the inclusion $\mathcal{B} \subseteq \mathcal{C}$ via β and γ; and
- for every i and j and every $f \in I_{ij}^{\mathfrak{A}}$, there exist substructures $\mathcal{B} \subseteq \mathcal{C}$ of $\mathcal{S}$ and isomorphisms $\beta : \mathcal{A}_i \twoheadrightarrow \mathcal{B}$ and $\gamma : \mathcal{A}_j \twoheadrightarrow \mathcal{C}$ such that f lifts to the inclusion $\mathcal{B} \subseteq \mathcal{C}$ via β and γ.

This cover is *uniformly computable* if $\mathfrak{A}$ is a uniformly computable simple cover of $\mathcal{S}$ and there exists a c.e. set W such that for all $i, j \in \mathbb{N}$,

$$I_{ij}^{\mathfrak{A}} = \{\varphi_e \restriction \mathcal{A}_i : \beta_3(e, i, j) \in W\}.$$

(Recall the computable bijection $\beta_3 : \mathbb{N}^3 \to \mathbb{N}$.) This says that a single process lists out all embeddings in each $\mathcal{I}_{ij}^{\mathfrak{A}}$ simulataneously.

A structure $\mathcal{B}$ is *locally computable* if it has a uniformly computable cover.

If $\mathfrak{A}$ is a computable simple cover, then every embedding of any $\mathcal{A}_i$ into any $\mathcal{A}_j$ is determined by its values on the generators of $\mathcal{A}_i$. Since $\mathcal{A}_i$ must be finitely generated, all such embeddings are computable, and therefore it is reasonable to call $\mathfrak{A}$ a computable cover without any further requirements on the sets $I_{ij}^{\mathfrak{A}}$. (Our main reason for considering only the finitely generated substructures of $\mathcal{S}$, rather than countable ones, is that embeddings among such structures are always computable.) For a uniformly computable cover, on the other hand, the sets $I_{ij}^{\mathfrak{A}}$ play a key role in our development of the subject, and it should be kept in mind that $I_{ij}^{\mathfrak{A}}$ need not contain every possible embedding of $\mathcal{A}_i$ into $\mathcal{A}_j$.

A cover feels like a category, and it almost is one. If the sets $I^{\mathfrak{A}}_{ii}$ always contain the identity map, and if $(g \circ f) \in I^{\mathfrak{A}}_{ik}$ for every $f \in I^{\mathfrak{A}}_{ij}$ and $g \in I^{\mathfrak{A}}_{jk}$, then the cover actually is a category. Many of the nicer covers can be closed under these conditions, and therefore can indeed be thought of as categories.

A substantial array of properties exists, which various covers do or do not satisfy. The field of real numbers has a uniformly computable cover, as defined above, but does not have one with any of these nice properties. The field of complex numbers, on the other hand, is *perfectly locally computable*: it has a uniformly computable cover satisfying the nicest known properties. So does every other algebraically closed field. So also does every computable structure, which is not surprising: a structure which is computable in the global sense certainly should have nice local-computability properties. Indeed, for countable structures, perfect local computability is equivalent to computable presentability.

The goal of local computability, however, is to consider uncountable structures $\mathcal{S}$, and to give computable (hence countable) descriptions of those structures. One way to do so is to produce computable structures highly similar to $\mathcal{S}$. Current work bu Mulcahey and the author has shown that every perfectly locally computable structure $\mathcal{S}$ has a computable *simulation* $\mathcal{C}$, i.e. a computable structure satisfying exactly the same definable properties as $\mathcal{S}$, and possessing exactly the same types of elements. For instance, if $\mathcal{S}$ is a perfectly locally computable field with an uncountable transcendence basis, and if $\mathcal{S}$ contains an algebraically closed subfield of transcendence degree 3 over $\mathbb{Q}$ but no such subfield of transcendence degree 4, then the same holds of $\mathcal{C}$, except that its transcendence basis is countable. In model-theoretic terms, $\mathcal{C}$ realizes exactly the same finitary types as $\mathcal{S}$, and indeed it can be made to do so over any finite parameter set from $\mathcal{S}$. (The fact that a transcendence basis is uncountable cannot be expressed in the finitary language of fields; the fact that it is infinite can be expressed by saying that $\mathcal{S}$ realizes a countable collection of finitary types, namely the n-type of n algebraically independent elements, for every $n \in \mathbb{N}$. Therefore the transcendence basis of $\mathcal{C}$ will also be infinite, but need not be uncountable – and cannot be, since $\mathcal{C}$ itself is countable.)

A perfectly locally computable structure therefore allows ordinary computable search procedures to be performed over any finite parameter set, such as the collection of coefficients of a polynomial. Consequently, it is hoped that many results about computable fields will carry over to perfectly locally computable fields, and likewise for computable differential fields. Lit-

tle about local computability is yet known; it has only been introduced very recently, and many basic questions remain open. Nevertheless, the author sees it as the best hope for applying the classical notion of Turing computation to uncountable structures, and in particular to uncountable fields and differential fields.

References

[1] C.C. Chang & H.J. Keisler, *Model Theory* (Amsterdam: Elsevier, 1990).

[2] H.M. Edwards, *Galois Theory* (New York: Springer-Verlag, 1984).

[3] M.D. Fried & M. Jarden, *Field Arithmetic* (Berlin: Springer-Verlag, 1986).

[4] A. Frohlich & J.C. Shepherdson; Effective procedures in field theory, *Phil. Trans. Royal Soc. London, Series A* **248** (1956) 950, 407-432.

[5] L. Harrington, Recursively presentable prime models, *Journal of Symbolic Logic* **39** (1974) 2, 305-309.

[6] W. Hodges; *A Shorter Model Theory* (Cambridge: Cambridge University Press, 1997).

[7] L. Kronecker; Grundzüge einer arithmetischen Theorie der algebraischen Größen, *J. f. Math.* **92** (1882), 1-122.

[8] M. Lerman, *Degrees of Unsolvability: Local and Global Theory* (Berlin: Springer-Verlag, 1983).

[9] G. Metakides & A. Nerode; Effective content of field theory, *Annals of Mathematical Logic* **17** (1979), 289-320.

[10] R.G. Miller; Computable fields and Galois theory, *Notices of the American Mathematical Society*, to appear.

[11] R.G. Miller; $\boldsymbol{d}$-computable categoricity for algebraic number fields, to appear.

[12] R.G. Miller, Locally computable structures, in *Computation and Logic in the Real World - Third Conference on Computability in Europe, CiE 2007*, eds. B. Cooper, B. Löwe, & A. Sorbi, *Lecture Notes in Computer Science* **4497** (Springer-Verlag: Berlin, 2007), 575-584. Also available at `qcpages.qc.cuny.edu/~rmiller/research.html`.

[13] M. Rabin; Computable algebra, general theory, and theory of computable fields, *Transactions of the American Mathematical Society* **95** (1960), 341-360.

[14] J.F. Ritt; *Differential Equations from the Algebraic Standpoint* (New York: American Mathematical Society, 1932).

[15] H. Rogers, Jr.; *Theory of Recursive Functions and Effective Computability* (New York: McGraw-Hill Book Co., 1967).

[16] R.I. Soare; *Recursively Enumerable Sets and Degrees* (New York: Springer-Verlag, 1987).

[17] V. Stoltenberg-Hansen & J.V. Tucker, Computable Rings and Fields, in *Handbook of Computability Theory*, ed. E.R. Griffor (Amsterdam: Elsevier, 1999), 363-447.

[18] A. Turing; On computable numbers, with an application to the Entscheidungsproblem, *Proceedings of the London Mathematical Society* (1936), 230-265.

[19] C. Wood, Differentially Closed Fields, in E. Bouscaren, ed., *Model Theory and Algebraic Geometry*, corrected second printing, *Lecture Notes in Mathematics 1696* (Berlin: Springer, 1999), 129-141.

Department of Mathematics
Queens College – C.U.N.Y.
65-30 Kissena Blvd.
Flushing, New York 11367 U.S.A.

Ph.D. Programs in Mathematics & Computer Science
The Graduate Center of C.U.N.Y.
365 Fifth Avenue
New York, New York 10016 U.S.A.

E-mail: `Russell.Miller@qc.cuny.edu`
Website: `qcpages.qc.cuny.edu/~rmiller`