Combinatory Differential Fields: An Algebraic Approach to Approximate Computation and Constructive Analysis [*]

Karl Aberer [†]

TR-91-061

November 1991

Abstract

The algebraic structure of *combinatory differential fields* is constructed to provide a semantics for computations in analysis. In this setting programs, approximations, limits and operations of analysis are represented as algebraic terms. Analytic algorithms can be derived by algebraic methods. The main tool in this construction are combinatory models which are inner algebras of Engeler graph models. As an universal domain of denotational semantics the lattice structure of the graph models allows to give a striking simple semantics for computations with approximations. As models of combinatory algebra they provide all essential computational constructs, including recursion. Combinatory models are constructed as extensions of first order theories. The classical first order theory to describe analysis is the theory of differential fields. It turns out that two types of computational constructs, namely composition and piecewise definition of functions, are preferably introduced as extensions of the differential fields theory. Combinatory differential fields are then the combinatory models of these enriched differential fields. We show for basic algorithms of computational analysis how their combinatory counterparts are derived in the algebraic setting. We illustrate how these algorithms are suitable to be implemented in a computer algebra environment like mathematica.

[*]Part of this work was done while the author was at ETH Zürich. Submitted to Journal of Symbolic Computation.

[†]International Computer Science Institute, Berkeley, CA 94704. email: aberer@icsi.berkeley.edu. Supported by *Schweizerische Gesellschaft zur Förderung der Informatik und ihrer Anwendungen*

1. Introduction

The first motivation that is basic for this work is to combine the power of *numeric computation* in solving analytic problems with the accuracy of *symbolic computation.* The additional power of numeric computations is mainly due to the fact that approximating algorithms may be used, which are potentially of an infinite form, while in symbolic computation manipulations are restricted to a finite number of steps, in which only provably exact results are computed. One exception of this are computations with closed-form representations of infinite power-series, e.g. for the solution of differential equations. This inspires a first approach contained in this work, to get an amalgamation of algebraic computation and computation of limits in analysis. Starting from the first order theory of differential fields, a standard way to formulate analysis in an algebraic form, we extend this theory axiomatically by operations representing programming constructs. This is in the spirit of symbolic computation where differential fields are extended by new transcendental functions in order to solve problems, just that we use now programs instead of transcendental functions. The approach works very well for representing composition and conditional functions, which leads to a natural extension of rational functions to piecewise rational functions. However for infinite recursion the only way to do this in first order theories is to accept *nonstandard models.* Although nonstandard models were already used in different contexts for the description of computations in first order theories [Richter & Szaeo, 1983, Jensen, 1972] we try to avoid these as many desirable algebraic relationships are lost.

To overcome these difficulties we have to include in a powerful computational model for analysis the concept of *approximation.* Approximations occur as incomplete information or knowledge about objects with which one computes or as the result of the imperfect execution of operations. The notion appears in many different contexts. Less explicitly as the notion of *error* in numeric computations [Linz, 1988], more explicitly in methods of *self-validating numerics* [Kaucher, 1983] like interval arithmetic or generalizations of it. The notion of the radius of information plays also a major role in *information-based complexity theory* [Traub *et al.*, 1988]. Computable approximations are a substantial part of *recursive analysis* [Pour-El & Richards, 1989]. Approximation appears also as a basic concept in *denotational semantics* and the use of models of denotational semantics, like complete partial orders (CPO's) or graph models, for modelling approximate computation in analysis was already proposed and conducted at different places [Fehlmann, 1981, Weihrauch, 1980]. What was missing in these models was an algebraic foundation which allows to build up an intimate relationship between algebraic relations and approximate algorithms by being able to construct approximate algorithms with algebraic methods. A model of denotational semantics that has an especially clear structure with respect to approximation as well to algebraization are *Engeler graph models* [Engeler, 1981A, Maeder, 1986]. Graph models were already used as programming semantics for several mathematical structures, like varieties, geometries and analysis [Engeler, 1981B, Engeler, 1984, Engeler, 1988, Fehlmann, 1981, Seeland, 1978]. We will show how to bring these graph models in a form where they allow the incorporation of the algebraic aspects of analysis as delivered by the differential fields theory. This goes back to an approach that was first outlined in [Engeler, 1990]. Then approximation and infinite recursion are supplied by the graph model and we are able to set up new algebraic relationships that are basic properties of approximations and infinite recursions.

Of course the scope of this work goes beyond the idea of algebraization of approximate computation. Combinatory differential fields can be understood as models for *constructive analysis*. Although many models of computations were devised, as *recursive analysis* in many variations, the theory of *machines on the reals* [Blum *et al.*, 1989] or *information-based complexity theory*, and many tight relationships exist between combinatory differential fields and the other approaches, the combination of the concepts of analysis, programming and approximation in an uniform structure is not found in any of the models mentioned above. So combinatory differential fields seem to be a natural basis to discuss *recursivity* and *complexity* in analysis. On the other hand the examples given in this work indicate that algorithms are formulated in combinatory differential fields in a very natural way with respect to implementation. They can provide the necessary structure to build a modern numeric-symbolic computing environment for analysis.

Overview

We start with an informal introduction to the subject. Without relying on hard definitions we present the essential ideas of the work. Then we discuss a major example, namely a combinatory Newton method, in which the impact of the new methodology can be recognized. To derive this method we make use of several basic properties of combinatory differential fields which are introduced in the informal part and will be proven in the following technical part.

The technical part introduces combinatory models in their greatest generality. Their essential properties are given. Among these the main theorem is of central importance, which tells that every combinatory model is an inner algebra of an Engeler graph model. Furthermore we discuss the basic properties of operations on approximations and recursion in the combinatory model.

To introduce combinatory differential fields we first axiomatize the composition and conditional operations, where the conditional operation is to represent decisions, as extensions of the theory of differential fields. We omit in this part the proofs for the main theorem which is a normal form result for polynomial terms containing composition and conditionals and refer to [Aberer, 1991]. We also shortly discuss the axiomatic approach for defining recursion. Next we introduce several notions which are of importance when discussing the solutions of problems. These allow to describe concepts like “rounding”, “approximate solution” or “almost equal” in a concise algebraic form. At the end we give a second example, how to solve linear differential equations in combinatory differential fields.

PART I. Introduction to Combinatory Differential Fields

Combinatory Models

Since in analytic algorithms approximations are often used and results of computations are obtained as limits of repeatedly refined approximations there is a considerable interest in formalizing these concepts in a computer theoretic view. The model proposed in this work is on the one hand clear in the information-theoretic concepts used for describing the computation with approximations and on the other hand gives a powerful mechanism for solving analytic problems in an algebraic setting.

The introductory part is intended to give a wide spectrum of readers the opportunity to inform themselves about the basic concepts of combinatory differential fields without going through all the technical details involved in the construction.

The first aspect necessary to describe approximate computations is the data *representation* of approximations. Typical examples coming from analysis are the following.

the distance between x_n and x is smaller than ε:	$\lvert x_n - x\rvert < \varepsilon$
the point x lies in the interval $[a, b]$:	$a \leq x \leq b$
the function f is at x^* almost zero:	$\lvert f(x^*)\rvert < \varepsilon$

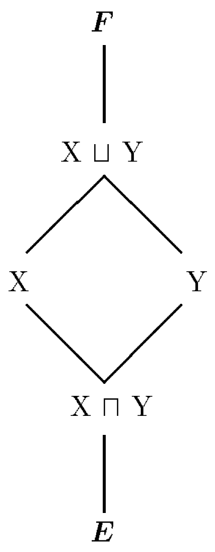
As one immediately recognizes information about approximations is in all three examples expressed in the form of *quantifier-free formulas* of first order predicate logic. The formulas used are built up using certain operational and predicate symbols. [1] We concentrate our interest in this work on real analysis. Since the real numbers are a totally ordered field we assume that among the operational symbols are at least the field operations $+, -, *$ and $^{-1}$ and there is besides equality $=$ a predicate symbol for total order $<$. Furthermore it may be appropriate to introduce other operation symbols, like $'$ for differentiation. Based on a logical language we now define approximations as follows: Assume we want to describe an object, a real number or a function, by its properties. Since we do not know what the object exactly is we give it some name, say @. Then express in terms of @ all the properties we know of this object by writing down formulas $\phi(@)$. For example if we know the object is strictly positive we write down the formula $@ > 0$. This gives a finite or infinite set of formulas. Then we identify the object with the information describing it

$$X = \{\phi_1(@), \phi_2(@), \ldots\}.$$

We denote the set of all possible formula-sets X by $A_@$. The degree of accuracy of the information describing such an object can widely vary. It can be precise, e.g. the real number 1 is described by the formula-set $\{@ = 1\}$ or it can be very vague, for example the formula-set $\{@ > 0\}$ is an appropriate description for all positive objects (including

[1] A collection of operational and predicate symbols and the rules how to form admissible terms and formulas out of the atomic symbols is called a *logical language*. In logic usually the operation symbols are called function symbols, but we avoid this terminology so that no confusion with the function notion of analysis occurs.

1). One recognizes an ordering structure on these formula-sets which is that of a *lattice.* Namely for any two formula-sets X, Y there exists another one which is satisfied by @ only if both X and Y are satisfied. The same way there exists also one which is satisfied by @ if at least one of X and Y is satisfied. The former is the supremum of X and Y and we denote it with $X \sqcup Y$ the latter the infimum which we denote with $X \sqcap Y$. For example if $X = \{\phi_1(@)\}$ and $Y = \{\phi_2(@)\}$ then $X \sqcup Y = \{\phi_1(@), \phi_2(@)\}$ and $X \sqcap Y = \{\phi_1(@) \vee \phi_2(@)\}$. Furthermore there exists a formula-set $\boldsymbol{E} = \{@ = @\}$ which is smaller than all others and one $\boldsymbol{F} = \{1 = 0\}$ which is the biggest possible. Thus we have not only a lattice but a *complete lattice.*



After giving a representation of approximation the next step is to perform operations on these approximations. The first basic computational step we consider is to perform the field operations, like addition. The question is, given two formula-sets X and Y, to define what is the sum $X + Y$ of them. Obviously the sum should again be an approximation described by a formula-set. The information contained in $X + Y$ should be everything that we can know about an object $x + y$, where x is satisfying X and y is satisfying Y. This means that the formula-set $X + Y$ should contain every formula $\phi(@)$, where $\phi(x + y)$ is a formula that is true under the assumption that x satisfies all properties of X and y those of Y. For example take the two formula-sets $\{@ = 1\}$ and $\{@ > 0\}$. Then the sum of an object where the first argument satisfies $\{@ = 1\}$ and the second satisfies $\{@ > 0\}$ obviously satisfies $\{@ > 1\}$. And this is already all information we can have about this new object. More formally we write this kind of operation down as follows.

$$X + Y = \{\phi(@) : X|_{@}^{x}, Y|_{@}^{y} \vdash \phi(x + y)\}.$$

The notation $X|_{@}^{x}$ means simply substitute every appearance of @ by x. The symbol $\vdash$ is to be read as "has as consequence" or "proves that". Hence in the example given this would read as follows

$$\{@ = 1\} + \{@ > 0\} = \{\phi(@) | x = 1, y > 0 \vdash \phi(x + y)\}.$$

This definition can be generalized to any k-ary operation $\tau(x_1, \ldots, x_k)$ that can be expressed as term of the logical language, like e.g. $\tau(x) = 1/2 * (x + 2/x)$ or $\tau(x, y, z) = x * (y + z)$ etc..

$$\boldsymbol{T}^{\tau(x_1,\ldots,x_k)}(X_1, \ldots, X_k) = \{\phi(@) : X_1|_{@}^{x_1}, \ldots, X_k|_{@}^{x_k} \vdash^T \phi(\tau(x_1, \ldots, x_k))\}$$

In principle many notions of consequence could be used at this point. In the technical development, we will later exactly specify which kind of consequence we mean, namely *logical consequence* under a first order *theory* T. [2] With respect to defining objects by their properties we make the following observation. Take the formula-set $X = \{@ > 1\}$. The objects satisfying this formula-set are exactly the same as those satisfying $X^* = \{@ > 0, @ > 1\}$. Hence X and X^* are equivalent in respect of expressing knowledge about objects. This is because $@ > 0$ is a logical consequence of $@ > 1$. Hence it makes no difference if we add to every formula-set all formulas which are logical consequences of this formula-set, or in other words if take the *logical closure*. We will call in the following such *logically closed* formula-sets *combinators* and denote the set of all combinators by $\mathcal{E}_{A_@}$. The set of all combinators also has the structure of a lattice. The operations defined for formula-sets can be understood as operations on combinators. It is clear from the definition that the result of such an operation is always logically closed and hence a combinator. The result of an operation also remains the same when logically closing the arguments before applying the operation.

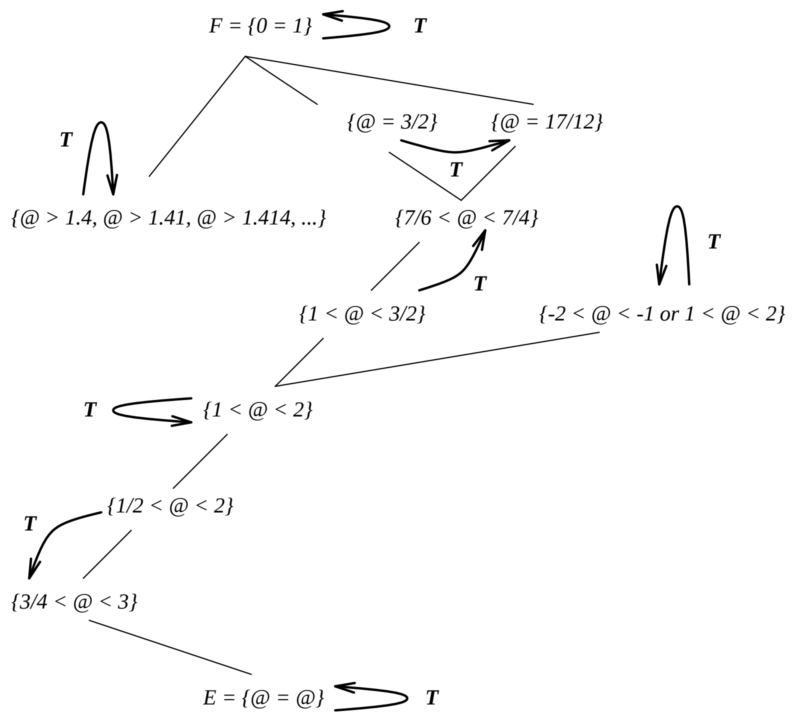
The following illustration shows a small part of the lattice of combinators that lie between $\boldsymbol{E}$ and $\boldsymbol{F}$. It also shows the possible effects of combinatory operations on combinators, in this case of the operation $\boldsymbol{T} = \boldsymbol{T}^{1/2*(x+2/x)}$. One recognizes that such an operation may refine knowledge, expand knowledge or leave it the same. In the last case the combinator is a fixed point of the operation. A typical phenomenon that may occur is that for a combinatory equation like the fixed point equation

$$X = \boldsymbol{T}^{1/2*(x+2/x)}(X)$$

many different solutions are possible. Among the fixed points in this particular example are $\boldsymbol{E}, \boldsymbol{F}, \{1 < @ < 2\}$ and $\{@ > 1.4, @ > 1.41, @ > 1.414, \ldots\}$. [3]

[2]The first order predicate calculus gives an exact mechanism for deriving new formulas out of given ones and a *theory* T is a set of sentences which are assumed to hold for the intended structures. So the field theory for example gives the set of properties we expect to hold in a field for the field operations and predicates. The notation $\vdash^T$ means, that the formulas on the right hand side can be derived from the formulas of the left hand side under the assumption that the sentences of T hold.

[3]Note that the recursively defined sequence $x_{n+1} := 1/2 * (x_n + 2/x_n)$ has $\sqrt{2}$ as limit when the starting value is appropriately chosen.



As already indicated, combinators are not necessarily finite objects. This is very useful in the context of analysis since this allows to describe real numbers and functions as limits of approximate computations. But as computation lives in a finite world we have to give a finite representation to these infinite objects. The idea here is to represent an infinite combinator by the union of an increasing chain $\boldsymbol{T}^0 \sqsubseteq \boldsymbol{T}^1 \sqsubseteq \boldsymbol{T}^2, \ldots$ of (finite) combinators which is recursively computed and hence has a finite representation by the program computing the sequence. For example for the infinite combinator $\{@ > 1.4, @ > 1.41, @ > 1.414, \ldots\}$ we can compute a representation by recursively computing the sequence

$$\boldsymbol{T}^{n+1} := \boldsymbol{T}^{1/2*(x+2/x)}(\boldsymbol{T}^n)$$

with $\boldsymbol{T}^0 = \{@ > 1.4\}$ and then taking the union

$$\bigsqcup_{n \in \mathbb{N}} \boldsymbol{T}^n.$$

More generally we now introduce such recursively defined combinators as a new operation on combinators defined as follows. Given a recursion

$$\begin{array}{rcl} \boldsymbol{T}^{m+1} & := & \boldsymbol{T}^m \sqcup \boldsymbol{T}(\boldsymbol{T}^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \\ \boldsymbol{T}_1^{m+1} & := & \boldsymbol{T}_1(\boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \\ & \vdots & \\ \boldsymbol{T}_k^{m+1} & := & \boldsymbol{T}_k(\boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \end{array}$$

with starting values

$$\boldsymbol{T}^0 = X, \boldsymbol{T}_1^0 = X_1, \ldots, \boldsymbol{T}_k^0 = X_k$$

we define the $k+1$-ary operation $\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k}$ on combinators by

$$\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k}(X,X_1,\ldots,X_k) = \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}^m.$$

The $\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k$ are operations on combinators which can be themselves recursions or defined in the basic way shown before.

Summarizing up to this point we have defined an *algebraic structure*

$$CMod(F) =< \mathcal{E}_{A_@};\ \boldsymbol{T}^{\tau(x_1,\ldots,x_k)}, \sqcup, \sqcap, \boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k} >$$

where the domain consists of all combinators of $\mathcal{E}_{A_@}$ and the algebraic expressions are built up by three types of operations:

operations defined by terms $\tau(x_1,\ldots,x_k)$,
lattice operations,
recursion operations.

These expressions can be understood as algebraic representations of programs. The details of this structure however depend strongly on the underlying theory T. Now the question is: what have we achieved?

First we have incorporated the notion of approximation in a computational model for analysis. Approximation is simply realized by subset inclusion of combinators, as it is given due to the lattice structure of $\mathcal{E}_{A_@}$. Thus a combinator X approximates Y if $X \sqsubseteq Y$. This means that an object, that has in principle an exact definition, like the solution of an equation, can be given an approximate form, which often is computationally more convenient.

The algebraic approach to the problem of computation with approximations allows the formal derivation of properties of combinators. This means we can make exact algebraic statements about objects which themselves are approximate descriptions of real numbers or functions. A typical example for this is the following fixed point property of unary recursions.

$$\boldsymbol{T}(\boldsymbol{M}^{\boldsymbol{T}}(\boldsymbol{T}^0)) = \boldsymbol{M}^{\boldsymbol{T}}(\boldsymbol{T}^0).$$

Typically such properties of recursion are due to *continuity* of the operations on combinators. Continuity is the property that for an ascending chain of combinators X_m such that $X_m \sqsubseteq X_{m+1}$ the following equality holds.

$$\boldsymbol{T}(\bigsqcup_{m\in\mathbb{N}} X_m) = \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}(X_m)$$

We will show that continuity is a fundamental property of all combinatory operations.

Another way to establish algebraic relationships between combinators is by lifting algebraic relations expressed in the underlying logic into the combinatory model. So if for example two terms τ_1 and τ_2 are given and we can prove in T that $\tau_1 = \tau_2$ then we can state that $\boldsymbol{T}^{\tau_1} = \boldsymbol{T}^{\tau_2}$. [4] But equality is not always obtained in this way. The lifting can be accompanied by some loss of information. So if in the theory there is $\tau_1(x,x) = \tau(x)$ where τ_1 is a term with two arguments, then in general in the combinatory model we can only show that

$$\boldsymbol{T}^{\tau_1(x_1,x_2)}(X,X) \sqsubseteq \boldsymbol{T}^{\tau(x)}(X).$$

We call this property *weakening.* Note that the inclusion still is to be understood as algebraic relation since we could reformulate $X \sqsubseteq Y$ as $X \sqcap Y = Y$. [5]

A way to look at combinatory models and combinatory differential fields, which are combinatory models of differential fields, is to view them as an extension mechanism, where problems which are not solvable in the ground field are solved in a richer structure. Classically in the theory of differential equations extensions are introduced as new functions which satisfy certain simple properties, as for example the exponential function. In our approach the new functions are introduced as programs, especially infinite recursions.

Up to now we have left relatively open what kind of theory is underlying the combinatory model. A first order theory that is basic for the symbolic treatment of analysis is the theory of differential fields. We will define combinatory differential fields essentially as combinatory models of the differential fields theory. Before doing this we want to enrich the theory of differential fields.

Algorithmic Extensions of Differential Fields

For the symbolic solution of differential equations and integrals, as well as for the solution of algebraic equations in differential fields, the notion of field extension is essential: a differential field D is extended by a new function f to the differential field $\bar{D} = D(f)$. This becomes necessary since e.g. most of the differential equations are not solvable over the ground field, which is typically a rational function field. For the solution of differential equations the typical extensions are the so-called *elementary extensions*, which are either

algebraic extensions: $p(f) = 0$, p a polynomial of D

or one of the following two types of *transcendental* extensions, which are defined as solutions of simple differential equations:

logarithmic extensions: $f' = g'/g$, $g \in D$, then $f = \log g$
exponential extensions: $f' = g' * f, g \in D$, then $f = \exp g$.

Choosing exactly this kind of extensions for the algebraic treatment of differential equations has obviously its justification in the fact that these extensions are easy to define and well understood. But as the algorithmic aspect is of central interest it also becomes natural to ask whether it is worth to analyze the possibility to extend differential fields by algorithmically defined functions.

[4] Note that the definition for combinatory operations also works for 0-ary operations.
[5] This is simply due to the fact that lattices are algebraic structures.

A differential field is typically a function field, e.g. $\mathbb{Q}(\iota)$, where ι denotes the identity function with the property $\iota(x) = x$. One basic construction on functions as well as on programs is *composition.* In traditional mathematical notation the composition of two functions $f(x)$ and $g(x)$ is denoted by $f(g(x))$. We denote composition in logic with a binary operation symbol and write for this $f \circ g$. The identity function ι then naturally satisfies $f \circ \iota = f = \iota \circ f$.

Many functions in mathematics are defined by cases. For example the absolute value function $|x|$ is defined as

$$|x| = \begin{cases} x, & x > 0 \\ -x, & x \leq 0 \end{cases}$$

We use the following general scheme to define functions by cases.

$$f(x) = \begin{cases} h_1(x), & g(x) > 0 \\ h_2(x), & g(x) \leq 0 \end{cases}$$

and denote this function f by

$$f = \text{cond } g \ h_1 \ h_2$$

These two operations are of considerable practical interest. The conditional operations allows to denote functions, e.g. the absolute value function, in closed form which otherwise would require quantified first order formulas for their description, and hence would be out of the range of description in combinatory models. We can also state using the composition function, a fundamental property of conditional functions, namely:

$$const(c) \wedge \sigma \circ c > 0 \rightarrow (cond\ \sigma\ \tau_1\ \tau_2) \circ c = \tau \circ c,$$

and similarly for $\sigma \circ c \leq 0$. From a theoretical point of view the conditional operation allows us to express every function $\mathbb{R} \rightarrow \mathbb{R}$ that can be computed in a finite number of steps and hence is a piecewise rational function, in closed form. In the following we will call the theory of differential fields extended by these two operations the theory of *conditional differential function fields*, or shortly $CDFF$. We define *combinatory differential fields* as the combinatory models of $CDFF$.

One could also ask whether it would not be of advantage to introduce in a similar way as it is done in combinatory models infinite recursion as an operation symbol of the logic. Consider as an example the exponential function, which cannot only be understood as a function defined as formal solution of the differential equation $y' = y$ but also as a recursively defined function by employing its power-series representation

$$\exp \iota = 1 + \frac{\iota}{1!} + \frac{\iota^2}{2!} + \frac{\iota^3}{3!} + \ldots .$$

This power-series can be computed by the following recursion

$$\begin{array}{rcll} y_{n+1} & = & y_n + u_n & y_0 = 0 \\ u_{n+1} & = & u_n * \frac{\iota}{k_n} & u_0 = 1 \\ k_{n+1} & = & k_n + 1 & k_0 = 1 \end{array}$$

We introduce the following notation in analogy to the combinatory model to denote the function defined by this recursion.

$$\mu_{y,u,k}^{\,y+u,\, u*\frac{\iota}{k},\, k+1}(0,1,1).$$

It turns out that there is no obvious way to extend a first order theory in this manner without introducing nonstandard models, i.e. allowing infinitesimal small and infinitely large elements. As a consequence, in contrast to the combinatory recursion operation, no fixed point property can be stated for recursion operators. Hence many possible algebraic relationships are lost. Thus, while we include logical representations of composition and conditionals in the combinatory differential fields, we will make no further use of the logical representation of recursion, but use the combinatory representation of recursion in combinatory differential fields.

Now let us turn to a concrete example to illustrate how this machinery works. We want to develop a combinatory version of Newton's method.

Combinatory Newton Method

Newton Iteration
The problem is to solve the equation

$$\gamma(y) = 0.$$

When we want to solve the problem only on certain domains (of the real axis) we can express this by using conditionals. So for the problem of computing the square root function in the positive reals we can state the problem as follows.

$$y^2 = cond\ \iota\ \iota\ 0.$$

Note that $cond\ \iota\ \iota\ 0$ is the identity function on the positive reals and constant zero on the negative reals. In conventional notation this reads as follows: find a function $y(x)$ such that $y^2(x) = x,\ x \geq 0$.

When there does not exist a closed form solution, which is in $CDFF$ by a piecewise rational function, we want to express a solution or approximations of it by combinatory terms. To do this we use recursive methods like Newton's method

$$y_{n+1} = y_n - \frac{\gamma(y_n)}{\gamma'(y_n)}.$$

Direct translation of Newton's method

As an example we consider in this paragraph the computation of the square root function on $[1, \infty]$. The problem can be stated by using conditional operations as

$$y^2 = cond\ (\iota - 1)\ \iota\ 0.$$

We discuss some of the problems and restrictions in the direct translation of Newton's method to a combinatory algorithm.

For applying Newton's method we have to choose a starting value close enough to the solution such that the iteration sequence converges to a zero. For example $y_0 = \iota$ would be a good choice. For applying the combinatory Newton method in order to obtain a monotone increasing chain of formula-sets we choose a function interval as initial approximation and starting value. Here a good choice is

$$X_0 = \{1 \leq_{[1,\infty]} @ \leq_{[1,\infty]} \iota\}$$

$\tau_1 \leq_{[a,b]} \tau_2$ is a short-hand notation for $a \leq c \leq b \rightarrow \tau_1 \circ c \leq \tau_2 \circ c$ which can be expressed in quantifier-free form by using conditionals. Newton's method gives rise to the following recursion.

$$y_{n+1} = y_n - \frac{y_n^2}{2 * y_n} = 1/2 * (y_n + \iota/y_n) := \chi(y_n).$$

Let the unary combinator $\boldsymbol{H}$ be the combinatory embedding of $\chi(x)$. We want to compute $X_{n+1} = \boldsymbol{H}(X_n)$. Since $\chi(1) = \chi(\iota) = 1 + \iota/2 \geq {}^+\sqrt{\iota} = \chi({}^+\sqrt{\iota})$ we have

$$X_n = \{\iota \leq_{[1,\infty]} @^2, @ \leq_{[1,\infty]} y_n, 0 \leq_{[1,\infty]} @\}$$

for $n = 1$. The computation furthermore shows that $X_0 \sqsubseteq X_1$. This is an important information as we can tell now, due to the fixed point property of recursion operators, that X_0 approximates a solution for $X = \boldsymbol{H}(X)$, namely $\boldsymbol{M}^{\boldsymbol{H}}(X_0) = \bigsqcup X_n$. The fact that y_n converges to the square root functions, which can be verified by numerical estimates, shows that this is an approximation converging to the square root function.

But there are some problems in connection with this "solution". First it contains in form of the inequality $\iota \leq_{[1,\infty]} @^2$ a problem that is equally difficult to solve as the original one. Second applying the operation $\boldsymbol{H}$ to a set of formulas can be a difficult mathematical and computational problem.

To avoid these difficulties we impose the following restrictions. First we restrict the formulas that may appear in a combinator to a certain type which is good to manage, e.g. the intervals. Second we allow only the use of a specific set of admissible operations for which we can easily compute the result for these simpler combinators.

In our example we choose function intervals for the representation of functions and allow only the field operations on those. Then define

$$\boldsymbol{H}_I(X) = 1/2 * (X + \frac{\boldsymbol{J}}{X}).$$

where we denote $\boldsymbol{J} = \{@ = \iota\}$. Then applying Newton's method to the starting value $X_0 = \{\alpha \leq_{[1,\infty]} @ \leq_{[1,\infty]} \beta\}$ gives the following result.

$$\boldsymbol{H}_I \cdot X_0 = 1/2 * (X_0 + \boldsymbol{J}/X_0) = \{1/2 * (\alpha + \iota/\beta) \leq_{[1,\infty]} @ \leq_{[1,\infty]} 1/2 * (\beta + \iota/\alpha)\}.$$

To satisfy the condition $X_0 \sqsubseteq \boldsymbol{H}_I \cdot X_0 = X_1$ such to ensure a solution is approximated by X_0 gives the condition $\alpha * \beta = \iota$. In this case $X_0 = \boldsymbol{H}_I \cdot X_0 = \boldsymbol{M}^{\boldsymbol{H}_I}(X_0)$. Hence this cannot give a convergent solution provided α, β are rational.

Let us summarize some of the difficulties experienced in directly translating the well known (numerical) algorithm into a combinatory algorithm:

- The embedding of a general operation $\tau(x_1, \ldots, x_n)$ can result in difficult computational problems.
- The liberal use of logical formulas can hide the problem in implicit form in the basic combinators.
- Putting restrictions on the straightforward algorithms to avoid the first two problems can make them useless.

So since we cannot change the first two points we have to look for better (combinatory) algorithms.

A Combinatory Algorithm

Let us return to the general problem of solving the equation

$$\gamma(x) = 0$$

Let $\boldsymbol{G}$ be a combinatory embedding of γ. We want to give a combinatory (recursive) algorithm which gives a convergent approximation of the solution of the corresponding combinatory problem

$$\boldsymbol{G}(X) = 0.$$

Furthermore this algorithm should still work when knowledge representation and admissible operations are restricted.

We do not make use of the derivative, as in Newton's method, but use instead, similarly as for the secant method, the difference quotient

$$\lambda_\gamma(x, y) := \frac{\gamma(x) - \gamma(y)}{x - y}.$$

We call this function the *Lipschitz function* since it represents exactly the contraction factor in the lipschitz condition

$$|\gamma(x) - \gamma(y)| = |\lambda_\gamma(x, y)| * |x - y|.$$

Note that when γ is a polynomial then $\lambda_\gamma(x,y)$ is also a polynomial. Let $\boldsymbol{L}$ be the binary operation representing the embedding of $\lambda_\gamma(x,y)$.

If we rewrite the definition of the lipschitz function as

$$x = y + \frac{\gamma(x) - \gamma(y)}{\lambda_\gamma(x,y)}$$

and apply weakening we obtain the following inclusion

$$Y + \frac{\boldsymbol{G}(X) - \boldsymbol{G}(Y)}{\boldsymbol{L}(X,Y)} \sqsubseteq X.$$

Let us denote with X^* a solution of the combinatory problem, for example the formal solution $X^* = \{@^2 = \mathit{cond}\ (\iota - 1)\ \iota\ 0\}$. Assume we know an approximation X_n to X^*, $X_n \sqsubseteq X^*$. If we take an arbitrary combinator y_n interpolating X_n, which means $X_n \sqsubseteq y_n$, [6] or in other words take a point y_n out of X_n, then according to the inclusion above, we can establish the following.

$$\begin{aligned} y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)} &\sqsubseteq \\ y_n + \frac{\boldsymbol{G}(X^*) - \boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)} &\sqsubseteq \\ y_n + \frac{\boldsymbol{G}(X^*) - \boldsymbol{G}(y_n)}{\boldsymbol{L}(X^*, y_n)} &\sqsubseteq X^*. \end{aligned}$$

The previous inclusions mean that if X_n is an approximation of the solution of $\boldsymbol{G}(X^*) = 0$ and y_n is taken out of X_n, and if we transform the information in y_n and X_n according to

$$y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)},$$

then we get further information approximating X^*.

Therefore this leads to a first version for an information refining algorithm for approximating the solution X^* of $\boldsymbol{G}(X) = 0$ starting with an initial approximation X_0 .

1) Choose y_n such that $X_n \sqsubseteq y_n$.

2) Compute $X_{n+1} = X_n \sqcup y_n - \dfrac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)}$

The algorithm given so for includes a random step, by allowing to choose any y_n such that $X_n \sqsubseteq y_n$. We can transform this step to a deterministic one if desired. Since $X_n \sqsubseteq y_n$ we conclude by using monotonicity, which is a general property of operations on combinators,

$$y_{n+1} = y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(y_n, y_n)} \sqsubseteq y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)} = X_{n+1}.$$

[6]We use the notion *interpolation* as the exact opposite of *approximation.*

This relation gives rise to the following nonmonotone recursion.

$$\begin{aligned} X_{n+1} &= X_n \sqcup y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(X_n, y_n)} \\ y_{n+1} &= y_n - \frac{\boldsymbol{G}(y_n)}{\boldsymbol{L}(y_n, y_n)}. \end{aligned}$$

We can summarize the previous arguments as follows.

Theorem 1 *If* $\boldsymbol{G}(X^*) = 0$ *and* $X_0 \sqsubseteq X^*, X_0 \sqsubseteq y_0$ *then*

$$\boldsymbol{M}_{X,y}^{\;y - \frac{\boldsymbol{G}(y)}{\boldsymbol{L}(X,y)},\, y - \frac{\boldsymbol{G}(y)}{\boldsymbol{L}(y,y)}}(X_0, y_0) = \bigsqcup_{n \in \mathbb{N}} X_n \sqsubseteq X^*.$$

Note that for proving this theorem we have used a few basic properties of operations on combinators, which will be given later in the theoretical treatment.

From the theorem we may also conclude that if the recursion results in the contradictionary set of formulas $\boldsymbol{F}$ no solution was contained in X_0.

The algorithm given is so general that it would be nearly more adequate to call it a Newton information transformation principle for fields. Let us point out some of the degree of freedom left open by the algorithm. The most important fact is that no assumptions are made on the type of knowledge used. It is of no importance whether one uses intervals, sets of intervals or other kinds of inequalities. (It seems less appropriate to use equalities with this algorithm.) The algorithm works for any kind of number or function field. In principle, γ can be any type of term, although the algorithm is up to now only successfully applied in the case where γ is a polynomial.

No statements are possible about the convergence rate of the algorithm in general, since this depends heavily on the concrete implementations and the type of knowledge involved. However we illustrate in the following two examples that convergence seems to be fast.

Computation of the Square Root Function

In the case where $\gamma(x) = x^2 - \iota$ the lipschitz function simplifies to $\lambda_\gamma(x, y) = x + y$. Hence the deterministic recursion is

$$\begin{aligned} X_{n+1} &= X_n \sqcup (y_n - \frac{y_n^2 - \iota}{X_n + y_n}) \\ y_{n+1} &= y_n - \frac{y_n^2 - \iota}{2 * y_n}. \end{aligned}$$

This method is similar to Moore's Newton iteration, a method known from interval arithmetic [Krawczyk, 1983]. We illustrate the convergence behaviour of this algorithm by a graph showing the result of this recursion computed in *mathematica*. In the graph two lines

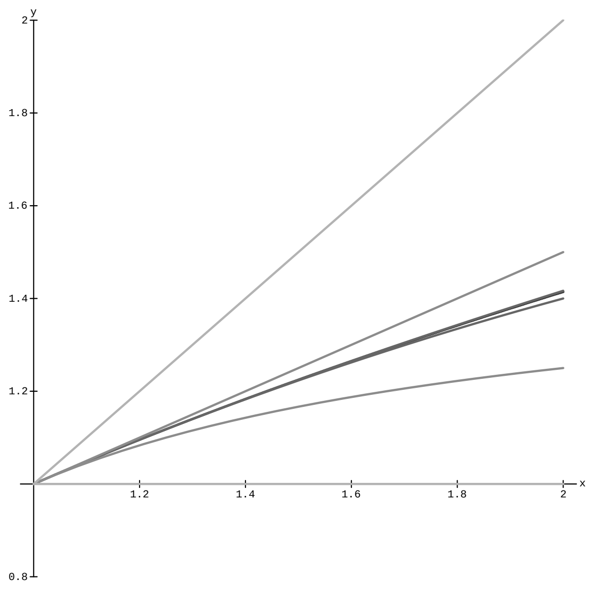


Figure 1: Three steps of recursion M

of the same darkness indicate the bounderies of the successive function intervals computed. Using a function interval arithmetic, which we have implemented in *mathematica* to carry out this example, the implementation of the algorithm is straightforward as may be taken from the program used.

```
F[0]={1, Int[1, J]}

F[{y_,X_}]:={Simplify[y-(y^2-J)/2/y], SimpInt[Cut[X, y-(y^2-J)/2/X]]}
```

The functions `Simplify` and `SimpInt` are used to simplify intermediate expressions. The first iterates are

```
Out[16]= {1, Int[1, J]}

          1 + J      -1 + 3 J  1 + J
Out[17]= {-----, Int[--------, -----]}
            2          2 J       2


                   2                     2    3             2
          1 + 6 J + J       -2 + 3 J + 8 J  - J   1 + 6 J + J
Out[18]= {------------, Int[--------------------, ------------]}
            4 + 4 J             4 (-1 + 3 J)       4 (1 + J)
```

The first component are the y_n, the second the X_n. The algorithm computes a monotonic decreasing sequence of rational function intervals.

Simultaneous Computation of the Real Zeros of a Polynomial

Now we apply the method to isolate the real zeros of a polynomial. We change the type of information used to *multi-intervals*, i.e. finite sets of intervals. To obtain convergence the nondeterministic version of the algorithm is used where y_{n+1} may be freely chosen. In a *randomized version* y_{n+1} would be chosen out of one of the intervals of X_n, while in a *parallelized version* for every interval in X_n a y_{n+1} is chosen and a new recursion started which can abort with an empty set (contradiction in the information) or contain a zero. Algorithms known in the literature which are related to this implementation of the combinatory Newton method are the method of derivatives, a symbolic algorithm, found in [Buchberger *et al.*, 1983] and an algorithm using successive linear interpolation (regula falsi) found in [Dekker, 1969].

The result of a (simulated) parallelized version again implemented in *mathematica* produced the following output. The number at the beginning denotes the depth in the parallel computation tree, `BotInt` is the empty interval which is described in the combinatory model by the combinator $\boldsymbol{F}$. Note that once an interval containing a zero is found the convergence appears to be quadratic.

```
Test[Int[-10.,10.], -6 + 11 y - 6 y^2 + y^3]

1 : MultInt[Int[-10., -0.0402685], Int[0.0350877, 10.]]
2 : Int[-3.79439, -0.0402685]
3 : Int[-1.12157, -0.0402685]
4 : BotInt
2 : MultInt[BotInt, Int[0.0350877, 4.76341]]
3 : MultInt[Int[0.0350877, 2.3763], Int[2.44174, 4.76341]]
4 : MultInt[Int[1.25383, 2.3763], Int[0.0350877, 1.1486]]
5 : MultInt[Int[1.86522, 2.3763], Int[1.25383, 1.65789]]
6 : Int[1.86522, 2.06164]
7 : Int[1.98948, 2.02338]
8 : Int[1.99953, 2.00041]
9 : Int[2., 2.]
Zero Int[2., 2.]
6 : MultInt[BotInt, BotInt]
5 : Int[0.772775, 1.1486]
6 : Int[0.987121, 1.02691]
7 : Int[0.999545, 1.00027]
8 : Int[1., 1.]
Zero Int[1., 1.]
4 : Int[2.58561, 3.41823]
5 : Int[2.9954, 3.0008]
6 : Int[3., 3.00001]
7 : Int[3., 3.]
Zero Int[3., 3.]
```

PART II. The Theory of Combinatory Differential Fields

Combinatory Models

In this section we give a rigorous algebraic basis to what we have developed on ideas about operations on knowledge so far. We show that operations on knowledge are elements of an inner algebra of an Engeler graph model of combinatory algebra. Thus we make the concept of knowledge transformation as well as that of infinite recursion to a subject of algebra.

We do this in a more general framework as exposed before. The following generalizations lead to no additional difficulties in the construction of the model.

- We define combinatory models not only for the theory of differential fields, but for any theory T.
- We generalize from knowledge about elements described by formula-sets consisting of formulas of the form $\phi(@)$, which is the same as describing unary relations by formulas, to knowledge about relations described by formulas of the form $\phi(@_1, \ldots, @_n)$, hence describing arbitrary relations.
- We define operations on knowledge not as the action of terms but as the action of a special class of first order formulas, the so called positive existential formulas. We will see later that operations defined by terms are just a special case of this.

Notations and Conventions

Let L be a language of first order logic with equality and T a theory in this language. Denote with $A_{@_1,\ldots,@_n}$ the set of quantifier-free formulas in L containing the additional constant symbols $@_1, \ldots, @_n$. We use subsets $X \subseteq A_{@_1,\ldots,@_n}$ to describe n-ary relations. The mapping $\boldsymbol{Cn} : \mathcal{P}(A_{@_1,\ldots,@_n}) \to \mathcal{P}(A_{@_1,\ldots,@_n})$ [7] defined by

$$\boldsymbol{Cn}(X) = \{\phi \in A_{@_1,\ldots,@_n} : X \vdash^T \phi\}$$

gives the *logical closure* of formula-sets in $A_{@_1,\ldots,@_n}$ under consequences of T. This does not change the extension of the relation described by X. Let $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ be the range of the mapping $\boldsymbol{Cn}$, hence the set of all logically closed formula-sets.

Now we want to define operations on the formula-sets. Each operation is defined by using a *positive existential formula.* A positive existential formula $\phi(x_1, \ldots, x_n, V_1, \ldots, V_k)$ with free variables $x_1, \ldots, x_n$ is built up from atomic formulas and the logical connectives $\wedge$, $\vee$, and $\exists$. Besides the usual atomic formulas of first order logic, namely equalities and predicates of

[7] $\mathcal{P}$ denotes the power set.

the logic, also *predicate variables* $V_i(t_1,\dots,t_m),\ i=1,\dots,k$ are allowed as atomic formulas, where $t_1,\dots,t_m$ are terms in L.

Definition of Combinatory Models
Let $X_1, X_2 \in \mathcal{E}_{A_{@_1,\dots,@_n}}$. For the union and intersection of logically closed formula-sets we introduce the following notations

$$\boldsymbol{Cn}(X_1 \cup X_2) = X_1 \sqcup X_2,\ \boldsymbol{Cn}(X_1 \cap X_2) = X_1 \sqcap X_2.$$

Note that that $\boldsymbol{Cn}(\boldsymbol{Cn}(X) \cap \boldsymbol{Cn}(Y)) = \boldsymbol{Cn}(X) \cap \boldsymbol{Cn}(Y)$.

For every positive existential formula ϕ we define inductively on the structure of ϕ for every arity $k \geq 0$ and $X_i \in A_{@_1,\dots,@_m},\ i=1,\dots,k$ mappings

$$\boldsymbol{T}^{\phi} : \mathcal{E}^k_{A_{@_1,\dots,@_m}} \to \mathcal{E}_{A_{@_1,\dots,@_n}}$$

$$\begin{array}{rcl}
\boldsymbol{T}^{\phi(x_1,\dots,x_n)}(X_1,\dots,X_k) & := & \boldsymbol{Cn}(\{\phi(@_1,\dots,@_n)\}) \\
\boldsymbol{T}^{V_i(t_1,\dots,t_m)}(X_1,\dots,X_k) & := & \boldsymbol{Cn}(X_i|^{t_1^*,\dots,t_m^*}_{@_1,\dots,@_m}),\ i \leq k \\
\boldsymbol{T}^{\phi\wedge\psi}(X_1,\dots,X_k) & := & \boldsymbol{T}^{\phi}(X_1,\dots,X_k) \sqcup \boldsymbol{T}^{\psi}(X_1,\dots,X_k)) \\
\boldsymbol{T}^{\phi\vee\psi}(X_1,\dots,X_k) & := & \boldsymbol{T}^{\phi}(X_1,\dots,X_k) \sqcap \boldsymbol{T}^{\psi}(X_1,\dots,X_k) \\
\boldsymbol{T}^{\exists x_i \phi(x_1,\dots,x_n,V_1,\dots,V_k)}(X_1,\dots,X_k) & := &
\end{array}$$

$$\{\psi(@_1,\dots,@_n) : \boldsymbol{T}^{\phi(x_1,\dots,x_n,V_1,\dots,V_k)}(X_1,\dots,X_k) \vdash^T \psi(x_1,\dots,x_{i-1},x_{i+1},\dots,x_n)\}$$

We used the notation $t_i^* := t_i|^{@_1,\dots,@_n}_{x_1,\dots,x_n}$.

Let $\boldsymbol{T}, \boldsymbol{T}_1,\dots,\boldsymbol{T}_k$ be mappings defined by positive existential formulas as above. Then we define a $k+1$-ary *recursion mapping*

$$\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\dots,\boldsymbol{T}_k} : \mathcal{E}^{k+1}_{A_{@_1,\dots,@_n}} \to \mathcal{E}_{A_{@_1,\dots,@_n}}$$

as

$$\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\dots,\boldsymbol{T}_k}(X, X_1,\dots,X_k) := \bigcup_{m\in\mathbb{N}} \boldsymbol{T}^m.$$

The $\boldsymbol{T}^m$ are computed by the following recursion.

$$\begin{array}{rcl}
\boldsymbol{T}^{m+1} & := & \boldsymbol{T}^m \sqcup \boldsymbol{T}(\boldsymbol{T}^m, \boldsymbol{T}^m_1,\dots,\boldsymbol{T}^m_k) \\
\boldsymbol{T}^{m+1}_1 & := & \boldsymbol{T}_1(\boldsymbol{T}^m_1,\dots,\boldsymbol{T}^m_k) \\
 & \vdots & \\
\boldsymbol{T}^{m+1}_k & := & \boldsymbol{T}_k(\boldsymbol{T}^m_1,\dots,\boldsymbol{T}^m_k)
\end{array}$$

with the starting values $\boldsymbol{T}^0 = X, \boldsymbol{T}^0_1 = X_1,\dots,\boldsymbol{T}^0_k = X_k$. Note that monotonicity of the recursion sequence is only required for the first component of the recursion sequence. [8]

[8]This is in contrast to the usual definition of higher-dimensional fixed point recursion in graph models but has turned out to be appropriate for the applications in analysis.

The *combinatory model* $\mathbf{E}_{A_{@_1,\ldots,@_n}}$ of the theory T is defined as the algebraic structure

$$\mathbf{E}_{A_{@_1,\ldots,@_n}} =< \mathcal{E}_{A_{@_1,\ldots,@_n}}; \boldsymbol{T}^{\phi}, \boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k} > .$$

Embedding in a Graph Model

In a next step we want to represent the operations constructed above as elements of an *inner algebra* of the *Engeler graph model* with $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ as carrier set. In this way we construct an algebraic structure extending the term-algebra given by the logic to an algebra for computing with properties. We carry here out the program for embedding differential fields into graph models proposed in [Engeler, 1990]. The basic definitions and properties of graph models can be found in [Engeler, 1981A].

The Engeler graph model is an algebraic structure $\mathbf{D} =< \mathcal{D}_A, \cdot >$ with one binary *application operation* $\cdot$ satisfying the laws of *combinatory algebra.* These are given by the following two axioms where $\boldsymbol{K}$ and $\boldsymbol{S}$ are two constant combinators.

$$\begin{aligned} \boldsymbol{K} \cdot X \cdot Y &= X \\ \boldsymbol{S} \cdot X \cdot Y \cdot Z &= X \cdot Z \cdot (Y \cdot Z) \end{aligned}$$

A fundamental property of this logic is *combinatory completeness* which says that every term $t(X_1, \ldots, X_n)$ is expressible in the form $\boldsymbol{T} \cdot X_1 \cdot \ldots \cdot X_n$ where $\boldsymbol{T}$ is a term in $\boldsymbol{K}$ and $\boldsymbol{S}$.

It is possible to give in combinatory logic combinators which correspond to basic computational constructs. The $\boldsymbol{B}$-combinator represents composition

$$\boldsymbol{B} \cdot X \cdot Y \cdot Z = X \cdot (Y \cdot Z).$$

The fixed point combinators $\boldsymbol{Y}$ solves fixed point equations by

$$X \cdot (\boldsymbol{Y} \cdot X) = \boldsymbol{Y} \cdot X.$$

Furthermore we can give representations $\underline{m}$ for natural numbers and combinators $\boldsymbol{N}, \boldsymbol{Z}$ and $\boldsymbol{Zero}$ that allow the representation of recursive programs on them.

$$\begin{aligned} &\boldsymbol{N} \cdot \underline{m} = \underline{m+1} \\ &\boldsymbol{Z} \cdot X = \underline{0} \end{aligned} \qquad \boldsymbol{Zero} \cdot \underline{m} \cdot X \cdot Y = \begin{cases} X, & m = 0 \\ Y, & m > 0 \end{cases}$$

The domain $\mathcal{D}_A$ of the graph model consists of all subsets of $G(A)$, the set of *arrow terms.* This set is given by the following inductive definition using a base set A.

$$\begin{aligned} G_0(A) &= A \\ G_{n+1}(A) &= G_n(A) \cup \{\alpha \rightarrow a : \alpha \text{ finite, }, \ \alpha \subseteq G_n(A), \ a \in G_n(A)\} \\ G(A) &= \bigcup_{n=0}^{\infty} G_n(A). \end{aligned}$$

The elements of $\mathcal{D}_A$ are called *combinators*. The application operation is defined for combinators $M, N \in \mathcal{D}_A$ by

$$M \cdot N = \{a : \exists\ \alpha \subseteq N,\ \alpha \to a \in M\}$$

The fact that the graph model is satisfying the laws of combinatory algebra is shown by giving representations of $\boldsymbol{S}$ and $\boldsymbol{K}$ in the graph model [Engeler, 1981A]. Furthermore the structure $\mathbf{D}_A$ is a complete continuous lattice [Maeder, 1986]. All combinatory operations are continuous, which means that for a given chain $X_1 \sqsubseteq X_2 \sqsubseteq \ldots$ and a combinator $\boldsymbol{T}$ the following holds:

$$\boldsymbol{T} \cdot \bigsqcup_{n \in \mathbb{N}} X_n = \bigsqcup_{n \in \mathbb{N}} \boldsymbol{T} \cdot X_n.$$

A *retraction* is a combinator $\boldsymbol{R}$ that satisfies $\boldsymbol{R} \cdot (\boldsymbol{R} \cdot X) = \boldsymbol{R} \cdot X$ for all $X \in \mathcal{D}_A$. An *inner algebra* $\mathbf{A} =< \mathcal{A}; \boldsymbol{T} >$ of a graph model has a retract $\mathcal{A} = \{X \in \mathcal{D}_A : \boldsymbol{R} \cdot X = X\}$ as *carrier set* and an operation $\boldsymbol{T} \in \mathcal{D}_A$ on this carrier set satisfies $\boldsymbol{T} \cdot (\boldsymbol{R} \cdot X_1) \cdot \ldots \cdot (\boldsymbol{R} \cdot X_n) = \boldsymbol{R} \cdot (\boldsymbol{T} \cdot (\boldsymbol{R} \cdot X_1) \cdot \ldots \cdot (\boldsymbol{R} \cdot X_n))$. Hence operations of the inner algebra are realized by application.

$$\boldsymbol{T}(X_1, \ldots, X_k) = \boldsymbol{T} \cdot X_1 \cdot \ldots \cdot X_k.$$

Now follows the main theorem which also justifies the use of the notion of combinatory models.

Theorem 2 *The combinatory model*

$$\mathbf{E}_{A_{@_1,\ldots,@_n}} =< \mathcal{E}_{A_{@_1,\ldots,@_n}}; \boldsymbol{T}^\phi, \boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k} >$$

of a theory T is an inner algebra of the graph model $\mathbf{D}_{A_{@_1,\ldots,@_n}}$.

Proof. $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ is the retract of $\boldsymbol{Cn}$. The operations of the combinatory model are by definition operations on the retract. So it is sufficient to show that the operations $\boldsymbol{Cn}$, $\boldsymbol{T}^\phi$ and $\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k}$ and compositions of those are representable as combinators of $\mathbf{D}_{A_{@_1,\ldots,@_n}}$.

First we give a combinatory representation of $\boldsymbol{Cn}$.

The following combinator is a retraction and satisfies due to the compactness theorem of first order logic $\boldsymbol{Cn} \cdot X = \boldsymbol{Cn}(X)$.

$$\boldsymbol{Cn} = \{\{\phi_1, \ldots, \phi_m\} \to \phi : \phi_1, \ldots, \phi_m \vdash^T \phi,\ \phi_1, \ldots, \phi_m, \phi \in A_{@_1,\ldots,@_n}\}$$

In the case of a k-ary mapping $\boldsymbol{T}^\phi$ the combinatory representation is given as follows.

Case ϕ free of V_i, atomic. Then take

$$\boldsymbol{T}^\phi = \boldsymbol{B}^k \cdot \boldsymbol{Cn} \cdot \{\emptyset \to \ldots \to \emptyset \to \phi(@_1, \ldots, @_n)\}$$

where the $\emptyset$ appears k times. $\boldsymbol{B}^k$ is the k-ary composition combinator of combinatory logic satisfying $\boldsymbol{B}^k \cdot Z \cdot X_1 \cdot \ldots \cdot X_k = Z \cdot (X_1 \cdot \ldots \cdot X_k)$.

Case $\phi = V_i(t_1, \ldots, t_m)$, $1 \leq i \leq k$. Then take

$$\boldsymbol{T}^{\phi} = \boldsymbol{B}^k \cdot \boldsymbol{Cn} \cdot \{\emptyset \to \ldots \to \psi(@_1, \ldots, @_m) \to \ldots \to \emptyset \to \psi(t_1^*, \ldots, t_m^*) : \psi \in A_{@_1,\ldots,@_n}\}$$

where $\psi(@_1, \ldots, @_m)$ is at the i-th place of the arrow term.

Case $\phi = \phi_1 \wedge \phi_2$ or $\phi = \phi_1 \vee \phi_2$. Union and intersection of sets of logical formulas closed under consequences is given in the graph model by the combinators

$$\begin{aligned} \boldsymbol{Un} &= \{\rho \to \psi \to \rho \wedge \psi : \rho, \psi \in A_{@_1,\ldots,@_n}\} \\ \boldsymbol{Is} &= \{\rho \to \psi \to \rho \vee \psi : \rho, \psi \in A_{@_1,\ldots,@_n}\}. \end{aligned}$$

Hence we have $X \sqcup Y = \boldsymbol{Un} \cdot X \cdot Y$ and $X \sqcap Y = \boldsymbol{Is} \cdot X \cdot Y$. By combinatory completeness we know that there exist combinators $\boldsymbol{S}^k$ with $\boldsymbol{S}^k \cdot Z_1 \cdot Z_2 \cdot X_1 \cdot \ldots \cdot X_k = Z_1 \cdot X_1 \cdot \ldots \cdot X_k \cdot (Z_2 \cdot X_1 \cdot \ldots \cdot X_k)$. Thus we get the following representation for conjunction and disjunction.

$$\begin{aligned} \boldsymbol{T}^{\phi_1 \wedge \phi_2} &= \boldsymbol{Un} \cdot \boldsymbol{S}^k \cdot \boldsymbol{T}^{\phi_1} \cdot \boldsymbol{T}^{\phi_2}, \\ \boldsymbol{T}^{\phi_1 \vee \phi_2} &= \boldsymbol{Is} \cdot \boldsymbol{S}^k \cdot \boldsymbol{T}^{\phi_1} \cdot \boldsymbol{T}^{\phi_2}. \end{aligned}$$

Case $\phi = \exists x_i\ \psi(x_1, \ldots, x_n, V_1, \ldots, V_k)$. We define the projection combinator

$$\begin{aligned} \boldsymbol{Pr}_n^i = \{\rho_1, \ldots, \rho_n \to \lambda\ : \rho_1, \ldots, \rho_n \vdash^T \lambda, \quad & \rho_1, \ldots, \rho_n \in A_{@_1,\ldots,@_n}, \\ & \lambda \in A_{@_1,\ldots,@_{i-1},@_{i+1},\ldots,@_n}\}. \end{aligned}$$

Note that $\boldsymbol{Pr}_n^i \cdot X = \boldsymbol{Cn} \cdot (\boldsymbol{Pr}_n^i \cdot X)$. Hence again referring to the compactness theorem of first order logic we get $\boldsymbol{T}^{\phi} \cdot (X_1 \cdot \ldots \cdot X_k) = \boldsymbol{Pr}_n^i \cdot (\boldsymbol{T}^{\psi(x_1,\ldots,x_n,V_1,\ldots,V_k)} \cdot (X_1 \cdot \ldots \cdot X_k))$, for all $X_1, \ldots, X_k \in \mathcal{E}_{A_{@_1,\ldots,@_m}}$. And so

$$\boldsymbol{T}^{\phi} = \boldsymbol{B}^k \cdot \boldsymbol{Pr}_n^i \cdot \boldsymbol{T}^{\psi(x_1,\ldots,x_n,V_1,\ldots,V_k)}$$

is the desired combinatorial representation.

To represent the nonmonotone recursion combinatorially we use the embedding of natural numbers in combinatory logic. Given a recursion

$$\begin{aligned} \boldsymbol{T}^{m+1} &:= \boldsymbol{T}^m \sqcup \boldsymbol{T}(\boldsymbol{T}^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \\ \boldsymbol{T}_1^{m+1} &:= \boldsymbol{T}_1(\boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \\ &\vdots \\ \boldsymbol{T}_k^{m+1} &:= \boldsymbol{T}_k(\boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_k^m) \end{aligned}$$

with starting values $\boldsymbol{T}^0 = X_0, \boldsymbol{T}_1^0 = X_1, \ldots, \boldsymbol{T}_k^0 = X_k$, where $X, X_1, \ldots, X_k$ are the arguments of the recursion combinator, we define first combinators $\boldsymbol{R}_0, \boldsymbol{R}_1, \ldots, \boldsymbol{R}_k$ such that

$$\boldsymbol{R}_0 \cdot \underline{m} = \boldsymbol{T}^m,\ \boldsymbol{R}_i \cdot \underline{m} = \boldsymbol{T}_i^m,\ i = 1, \ldots, k.$$

Define auxiliary combinators $\boldsymbol{F}_i,\ i = 0, \ldots, k$ as follows.

$$\begin{aligned}&\boldsymbol{F}_0 \cdot Z_0 \cdot Z_1 \cdot \ldots \cdot Z_k \cdot \underline{m} = \\ &\boldsymbol{Zero} \cdot \underline{m} \cdot \boldsymbol{T}^0 \cdot (\boldsymbol{Un} \cdot (Z_0 \cdot \underline{m-1}) \cdot (\boldsymbol{T} \cdot (Z_0 \cdot \underline{m-1}) \cdot (Z_1 \cdot \underline{m-1}) \cdot \ldots \cdot (Z_k \cdot \underline{m-1}))),\end{aligned}$$

$$\boldsymbol{F}_i \cdot Z_0 \cdot Z_1 \cdot \ldots \cdot Z_k \cdot \underline{m} = \boldsymbol{Zero} \cdot \underline{m} \cdot \boldsymbol{T}_i^0 \cdot (\boldsymbol{T}_i \cdot (Z_1 \cdot \underline{m-1}) \cdot \ldots \cdot (Z_k \cdot \underline{m-1})),\ i > 0.$$

Then, using the fixed point combinators $\boldsymbol{Y}_i^{k+1}, i = 1, \ldots, k+1$ of combinatory algebra, we define $\boldsymbol{R}_i, i = 0, \ldots, k$ as the least fixed points of the system $X_i = \boldsymbol{F}_i \cdot X_0 \cdot \ldots \cdot X_k,\ i = 0, \ldots, k$ by

$$\boldsymbol{R}_i = \boldsymbol{Y}_{i+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k,\ i = 0, \ldots, k.$$

We prove the property stated for $\boldsymbol{R}_i$ by induction over m.

For the case $m = 0$ and $i > 0$ we have.

$$\begin{aligned}\boldsymbol{R}_i \cdot \underline{0} &= \boldsymbol{Y}_{i+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k \cdot \underline{0} \\ &= \boldsymbol{F}_i \cdot (\boldsymbol{Y}_0^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k) \cdot \ldots \cdot (\boldsymbol{Y}_{k+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k) \cdot \underline{0} = \boldsymbol{T}_i^0.\end{aligned}$$

and similarly for $\boldsymbol{R}_0 \cdot \underline{0}$. For the induction step and $i > 0$ we have

$$\begin{aligned}\boldsymbol{R}_i \cdot \underline{m+1} &= (\boldsymbol{Y}_{i+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k) \cdot \underline{m+1} \\ &= \boldsymbol{F}_i \cdot (\boldsymbol{Y}_0^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k) \cdot \ldots \cdot (\boldsymbol{Y}_{k+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k) \cdot \underline{m+1} \\ &= \boldsymbol{T}_i \cdot (\boldsymbol{Y}_0^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k \cdot \underline{m}) \cdot \ldots \cdot (\boldsymbol{Y}_{k+1}^{k+1} \cdot \boldsymbol{F}_0 \cdot \boldsymbol{F}_1 \cdot \ldots \cdot \boldsymbol{F}_k \cdot \underline{m}) \\ &= \boldsymbol{T}_i \cdot (\boldsymbol{R}_0 \cdot \underline{m}) \cdot \ldots \cdot (\boldsymbol{R}_k \cdot \underline{m}) = \boldsymbol{T}_i^{m+1}.\end{aligned}$$

Again for $\boldsymbol{R}_0 \cdot \underline{m+1}$ we proceed similarly.

The fixed point $\boldsymbol{Y} \cdot \boldsymbol{N}^*$ of the combinator

$$\boldsymbol{N}^* \cdot X = \underline{0} \sqcup X \sqcup \boldsymbol{N} \cdot X$$

is the union of all representations of natural numbers: $\boldsymbol{Y} \cdot \boldsymbol{N}^* = \bigsqcup_{m \in \mathbb{N}} \underline{m}$. Hence

$$\boldsymbol{R}_0 \cdot (\boldsymbol{Y} \cdot \boldsymbol{N}^*) = \boldsymbol{R}_0 \cdot \bigsqcup_{m \in \mathbb{N}} \underline{m} = \bigsqcup_{m \in \mathbb{N}} \boldsymbol{R}_0 \cdot \underline{m} = \bigsqcup_{n \in \mathbb{N}} \boldsymbol{T}^m$$

gives the combinatorial representation of recursion.

Note that using continuity the following holds

$$\boldsymbol{Cn} \cdot \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T}^m = \bigsqcup_{m \in \mathbb{N}} \boldsymbol{Cn} \cdot \boldsymbol{T}^m = \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T}^m$$

This shows that the result of the recursion is logically closed.

After having given combinatorial representations for all operations we have only to remark that compositions of operations can be represented by using the $\boldsymbol{B}^k$ combinators. Thus we have represented the combinatory models as inner algebra of the graph model. ∎

Remark.

1. The domain $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ of the inner algebra is a complete lattice with an infimum denoted by $\boldsymbol{E} = \boldsymbol{Cn} \cdot \emptyset$ and a supremum denoted by $\boldsymbol{F} = \boldsymbol{Cn} \cdot \{\phi\}$ where ϕ can be any contradiction, like $0 = 1$. [9] We have the property that for $X_1, \ldots, X_k \in \mathcal{E}_{A_{@_1,\ldots,@_n}}$

$$\boldsymbol{T} \cdot X_1 \cdot \ldots \cdot X_k = \boldsymbol{F}, \text{ if exists } i,\ X_i = \boldsymbol{F}$$

The operators of the algebra again form a lattice.

2. The combinators $\boldsymbol{T}^{V_i(x_1,\ldots,x_n)}$ provide *projections* as follows, $X_i \in \mathcal{E}_{A_{@_1,\ldots,@_n}}$.

$$\boldsymbol{T}^{V_i(x_1,\ldots,x_n)} \cdot X_1 \cdot \ldots \cdot X_k = X_i$$

3. Union and intersection are represented by

$$\begin{aligned} \boldsymbol{T}^{V_1(x_1,\ldots,x_n) \wedge V_2(x_1,\ldots,x_n)} \cdot X_1 \cdot X_2 = X_1 \sqcup X_2 \\ \boldsymbol{T}^{V_1(x_1,\ldots,x_n) \vee V_2(x_1,\ldots,x_n)} \cdot X_1 \cdot X_2 = X_1 \sqcap X_2 \end{aligned}$$

Internal Combinatory Models

We come now to an important subclass of operations which are defined as the action of terms on formula-sets. Denote with $Te(x_1, \ldots, x_k)$ the set of terms of L with free variables $x_1, \ldots, x_k$. Let a term $\tau(x_1, \ldots, x_k)$ be given. Then we define a mapping $\boldsymbol{T}^{\tau(x_1,\ldots,x_k)} : \mathcal{E}^k_{A_{@_1}} \to \mathcal{E}_{A_{@_1}}$ as combinatory operation by $\boldsymbol{T}^{\tau(x_1,\ldots,x_k)} := \boldsymbol{T}^{\phi(x_1,V_1,\ldots,V_k)}$ for

$$\phi(x_1, V_1, \ldots, V_k) :\equiv \exists x_2, \ldots, x_{k+1}(x_1 = \tau(x_2, \ldots, x_{k+1}) \wedge V_1(x_2) \wedge \ldots \wedge V_k(x_{k+1})).$$

Combinators of this kind are called *internal combinators.* In many practical applications it will be sufficient to consider the algebraic structure which consists of combinators representing unary relations and the internal combinatory operations on them. In this case we speak of *internal combinatory models* [10] which are the algebraic structures

$$< \mathcal{E}_{A_@}; \boldsymbol{T}^{\tau}, \sqcup, \sqcap, \boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k} > .$$

Remark. In the case where a variable-free term defines an operation we get

$$\boldsymbol{T}^{\tau} := \boldsymbol{T}^{x_1=\tau} = \boldsymbol{Cn} \cdot \{x_1 = \tau\}.$$

[9] As a mnemonic: $\boldsymbol{E} = empty$, $\boldsymbol{F} = full$ set of properties.

[10] In the informal part we have only discussed internal combinatory models.

Sometimes we will also consider the following generalization of unary internal combinators to the embedding of a *vector operation.* Let a vector of terms $\vec{\tau}(x_1,\ldots,x_m)$ of length n be given.

$$\begin{aligned}\phi(x_1,\ldots,x_n,V_1) :\equiv \exists x_{n+1}\ldots x_{n+m} \quad & x_1 = \tau_1(x_{n+1},\ldots,x_{n+m}) \wedge \ldots \\ & \wedge x_n = \tau_k(x_{n+1},\ldots,x_{n+m}) \wedge V_1(x_{n+1},\ldots,x_{n+m}).\end{aligned}$$

This is a mapping $\mathcal{E}_{A_{@_1,\ldots,@_m}} \to \mathcal{E}_{A_{@_1,\ldots,@_n}}$. This definition cannot be extended in a natural way to a k-ary combinator.

We give now some of the basic properties of internal combinators. Before doing this we need some preparations from logic. The first is concerning notation: Whenever we write $\tau_1(x_1,\ldots,x_k) = \tau_2(x_1,\ldots,x_n)$ it has to be understood as

$$\vdash^T \forall x_1 \ldots x_k \tau_1(x_1,\ldots,x_k) = \tau_2(x_1,\ldots,x_k)$$

The second is a simple property

Lemma 1 *Let T be a theory, @ a constant symbol not appearing in T, τ a variable-free term. Then*

$$T, @ = \tau \vdash \phi(@) \quad \textit{iff} \quad T \vdash \phi(\tau).$$

First we give a characterization of internal combinators that relate these to the definition given in the introductory part.

Proposition 1 (Characterization of internal combinators) *Let $\tau \in Te(x_1,\ldots,x_k)$, $k \geq 1$, $X_1,\ldots,X_k \in \mathcal{E}_{A_{@_1}}$. Then*

$$\boldsymbol{T}^{\tau(x_1,\ldots,x_k)} \cdot X_1 \cdot \ldots \cdot X_k = \{\phi(@_1) : X_1|^{x_1}_{@_1},\ldots,X_k|^{x_k}_{@_1} \vdash^T \phi(\tau(x_1,\ldots,x_k))\}$$

Proof.

$$\begin{aligned}
&\boldsymbol{T}^{\exists x_2,\ldots,x_{k+1}(x_1=\tau(x_2,\ldots,x_{k+1})\wedge V_1(x_2)\wedge\ldots\wedge V_k(x_{k+1}))} \cdot X_1 \cdot \ldots \cdot X_k = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots \cdot (\boldsymbol{Pr}^{k+1}_{k+1} \cdot \boldsymbol{T}^{x_1=\tau(x_2,\ldots,x_{k+1})\wedge V_1(x_2)\wedge\ldots\wedge V_k(x_{k+1})} \cdot X_1 \cdot \ldots \cdot X_k)\ldots) = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots \cdot (\boldsymbol{Pr}^{k+1}_{k+1} \cdot \boldsymbol{T}^{x_1=\tau(x_2,\ldots,x_{k+1})} \cdot X_1 \cdot \ldots \cdot X_k \sqcup \\
&\qquad \boldsymbol{T}^{V_1(x_2)} \cdot X_1 \cdot \ldots \cdot X_k \sqcup \ldots \sqcup \boldsymbol{T}^{V_k(x_{k+1})} \cdot X_1 \cdot \ldots \cdot X_k))\ldots) = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots \cdot (\boldsymbol{Pr}^{k+1}_{k+1} \cdot (\{@_1 = \tau(@_2,\ldots,@_{k+1})\} \sqcup X_1|^{@_2}_{@_1} \sqcup \ldots \sqcup X_k|^{@_{k+1}}_{@_k}))\ldots) = \\
&\{\phi(@_1) : @_1 = \tau(@_2,\ldots,@_{k+1}), X_1|^{@_2}_{@_1},\ldots,X_k|^{@_{k+1}}_{@_1} \vdash^T \phi(@_1)\} = \\
&\{\phi(@_1) : X_1|^{@_2}_{@_1},\ldots,X_k|^{@_{k+1}}_{@_1} \vdash^T \phi(\tau(@_2,\ldots,@_{k+1}))\}
\end{aligned}$$

The last step is by using lemma 1. ∎

Remark. A similar characterization can be given for the embedding of vector operations. Let $\vec{\tau}(x_1,\ldots,x_m)$ be a vector of terms in $Te^n(x_1,\ldots,x_m)$ of length n. Let $X \in \mathcal{E}_{A_{@_1,\ldots,@_m}}$. Then the embedding of $\vec{\tau}$ is given by

$$\boldsymbol{T}^{\vec{\tau}(x_1,\ldots,x_m)} \cdot X = \{\phi(@_1,\ldots,@_m) : X|^{x_1,\ldots,x_m}_{@_1,\ldots,@_m} \vdash^T \phi(\tau_1(x_1,\ldots,x_m),\ldots,\tau_n(x_1,\ldots,x_m))\}.$$

Proposition 2 (Soundness of the Embedding) *Let $\tau_1, \tau_2 \in Te(x_1,\ldots,x_k)$. Then*

$$\tau_1(x_1,\ldots,x_k) = \tau_2(x_1,\ldots,x_k) \rightarrow \boldsymbol{T}^{\tau_1(x_1,\ldots,x_k)} = \boldsymbol{T}^{\tau_2(x_1,\ldots,x_k)}$$

Proof. We have $\boldsymbol{T}^{x_1=\tau_1(x_2,\ldots,x_{k+1})} = \boldsymbol{T}^{x_1=\tau_2(x_1,\ldots,x_{k+1})}$ since $\vdash^T x_1 = \tau_1(x_2,\ldots,x_{k+1}) \leftrightarrow x_1 = \tau_2(x_1,\ldots,x_{k+1})$. Hence inspecting the definition of the combinatory operations the combinators $\boldsymbol{T}^{\tau_1(x_1,\ldots,x_k)}$ and $\boldsymbol{T}^{\tau_2(x_1,\ldots,x_k)}$ are built up the same way. ∎

The converse of this theorem which is called the *completeness property* does not hold in general. However using the following criterion completeness can be shown for many term classes.

Criterion: Let a term class be given such that for two terms $\tau_1(x_1,\ldots,x_k), \tau_2(x_1,\ldots,x_k)$ the equality $\tau_1(x_1,\ldots,x_k) = \tau_2(x_1,\ldots,x_k)$ is decidable for T. Furthermore assume it is possible to give whenever $\tau_1(x_1,\ldots,x_k) \neq \tau_2(x_1,\ldots,x_k)$ variable-free terms $\gamma_1,\ldots,\gamma_k$ such that $\tau_1(\gamma_1,\ldots,\gamma_k) \neq \tau_2(\gamma_1,\ldots,\gamma_k)$. Then completeness holds for this term class.

The next proposition shows that the term algebra of L is isomorphically represented by the inner algebra with the carrier set $\mathcal{T}^0 = \{\boldsymbol{T}^\tau, \tau$ variable-free term of L $\}$ and the internal operations $\mathcal{T}^k = \{\boldsymbol{T}^{\tau(x_1,\ldots,x_k)} : \tau(x_1,\ldots,x_k) \in Te(x_1,\ldots,x_k)\}$.

Proposition 3 (Compositionality) *Let $\tau_1,\ldots,\tau_k$ be variable-free. Then*

$$\boldsymbol{T}^{\tau(\tau_1,\ldots,\tau_k)} = \boldsymbol{T}^{\tau(x_1,\ldots,x_k)} \cdot \boldsymbol{T}^{\tau_1} \cdot \ldots \cdot \boldsymbol{T}^{\tau_k}.$$

Proof.

$$\begin{aligned}
&\boldsymbol{T}^{\exists x_2,\ldots,x_{k+1}(x_1=\tau(x_2,\ldots,x_{k+1})\wedge V_1(x_2)\wedge\ldots\wedge V_k(x_{k+1}))} \cdot \boldsymbol{T}^{\tau_1} \cdot \ldots \cdot \boldsymbol{T}^{\tau_k} = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots (\boldsymbol{Pr}^{k+1}_{k+1} \cdot (\boldsymbol{T}^{x_1=\tau(x_2,\ldots,x_{k+1})} \cdot \boldsymbol{T}^{\tau_1} \cdot \ldots \cdot \boldsymbol{T}^{\tau_k} \sqcup \\
&\qquad \boldsymbol{T}^{V_1(x_2)} \cdot \boldsymbol{T}^{\tau_1} \cdot \ldots \cdot \boldsymbol{T}^{\tau_k} \sqcup \ldots \sqcup \boldsymbol{T}^{V_k(x_{k+1})} \cdot \boldsymbol{T}^{\tau_1} \cdot \ldots \cdot \boldsymbol{T}^{\tau_k})) \ldots) = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots (\boldsymbol{Pr}^{k+1}_{k+1} \cdot (\{@_1 = \tau(@_2,\ldots,@_{k+1})\} \sqcup \{@_2 = \tau_1\} \sqcup \ldots \sqcup \{@_{k+1} = \tau_k\})) \ldots) = \\
&\boldsymbol{Pr}^2_{k+1} \cdot (\ldots (\boldsymbol{Pr}^{k+1}_{k+1} \cdot (\{@_1 = \tau(\tau_1,\ldots,\tau_k)\} \sqcup \{@_2 = \tau_1\} \sqcup \ldots \sqcup \{@_{k+1} = \tau_k\})) \ldots) = \\
&\boldsymbol{Cn} \cdot \{@_1 = \tau(\tau_1,\ldots,\tau_k)\}
\end{aligned}$$

The last step is by repeated application of lemma 1. ∎

Remark. In the general case compositionality holds no longer but for example weakenings as the following may occur.

$$\boldsymbol{T}^{\tau(x_1,\dots,x_k)} \cdot (\boldsymbol{T}^{\tau_1(x)} \cdot X) \cdot \ldots \cdot (\boldsymbol{T}^{\tau_k(x)} \cdot X) \sqsubseteq \boldsymbol{T}^{\tau(\tau_1(x),\dots,\tau_k(x))} \cdot X.$$

We show now a very basic kind of weakening.

Proposition 4 (Weakening) *Let* $\tau_1(x_1,\dots,x_k) = \tau_2(x_1,\dots,x_1,\dots,x_k,\dots,x_k)$ *and* $X_1,\dots,X_k \in \mathcal{E}_{A_{@_1}}$. *Then*

$$\boldsymbol{T}^{\tau_2(x_1,\dots,x_1,\dots,x_k,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_1 \cdot \ldots \cdot X_k \cdot \ldots \cdot X_k \sqsubseteq \boldsymbol{T}^{\tau_1(x_1,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_k$$

Proof. Assume $\phi(@_1) \in \boldsymbol{T}^{\tau_2(x_1,\dots,x_1,\dots,x_k,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_1 \cdot \ldots \cdot X_k \cdot \ldots \cdot X_k$. Then

$$X_1|^{x_{11}}_{@_1},\dots,X_1|^{x_{1n_1}}_{@_1},\dots,X_k|^{x_{k1}}_{@_1},\dots,X_k|^{x_{kn_k}}_{@_1} \vdash^T \phi(\tau_2(x_{11},\dots,x_{1n_1},\dots,x_{k1},\dots,x_{kn_k})).$$

Therefore also

$$X_1|^{x_1}_{@_1},\dots,X_1|^{x_1}_{@_1},\dots,X_k|^{x_k}_{@_1},\dots,X_k|^{x_k}_{@_1} \vdash^T \phi(\tau_2(x_1,\dots,x_1,\dots,x_k,\dots,x_k)),$$

and so $X_1|^{x_1}_{@_1},\dots,X_k|^{x_k}_{@_1} \vdash^T \phi(\tau_1(x_1,\dots,x_k))$. This shows that

$$\phi(@_1) \in \boldsymbol{T}^{\tau_1(x_1,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_k.$$

∎

In the following situation equality is obtained.

Proposition 5 *Let* $\tau_1(x_1,\dots,x_k) = \tau_2(x_1,\dots,x_k)$ *where each variable in both terms appears exactly once. Then*

$$\boldsymbol{T}^{\tau_1(x_1,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_k = \boldsymbol{T}^{\tau_2(x_1,\dots,x_k)} \cdot X_1 \cdot \ldots \cdot X_k.$$

Proof. The proof is the same as in the previous proposition. But in this case it is possible to proceed in both directions. ∎

The following proposition holds only for the embedding of unary operations respectively vector operations.

Proposition 6 (Unary Characterization) *Let* $\vec{\tau}(x_1,\dots,x_m)$ *be in* $Te^n(x_1,\dots,x_m)$.

$$\phi(\tau_1(@_1,\dots,@_m),\dots,\tau_n(@_1,\dots,@_m)) \in X$$
$$\textit{iff}$$
$$\phi(@_1,\dots,@_n) \in \boldsymbol{T}^{(\tau_1(x_1,\dots,x_m),\dots,\tau_n(x_1,\dots,x_m))} \cdot X$$

Proof.

$$\begin{aligned}
&\phi(\tau_1(@_1,\ldots,@_m),\ldots,\tau_n(@_1,\ldots,@_m)) \in X \text{ iff}\\
&X|^{x_1,\ldots,x_m}_{@_1,\ldots,@_m} \vdash^T \phi(\tau_1(x_1,\ldots,x_m),\ldots,\tau_n(x_1,\ldots,x_m)) \text{ iff}\\
&\phi(\tau_1(x_1,\ldots,x_m),\ldots,\tau_n(x_1,\ldots,x_m)) \in \boldsymbol{Cn}\cdot X = X
\end{aligned}$$

▮

The unary characterization lemma can be used to show the following properties of internal combinators which are generalizations of compositionality to terms containing free variables.

Proposition 7 *Let $\tau(x), \tau_i(x) \in Te(x), i = 1,\ldots,k$ and $\sigma(x_1,\ldots,x_k) \in Te(x_1,\ldots,x_k)$ be given and $X_1,\ldots,X_k \in \mathcal{E}_{A_@}$. Then*

$$\begin{aligned}
\boldsymbol{T}^{\tau(x)}\cdot(\boldsymbol{T}^{\sigma(x_1,\ldots,x_k)}\cdot X_1\cdot\ldots\cdot X_k) &= \boldsymbol{T}^{\tau(\sigma(x_1,\ldots,x_k))}\cdot X_1\cdot\ldots\cdot X_k\\
\boldsymbol{T}^{\sigma(x_1,\ldots,x_k)}\cdot(\boldsymbol{T}^{\tau_1(x)}\cdot X_1)\cdot\ldots\cdot(\boldsymbol{T}^{\tau_k(x)}\cdot X_k) &= \boldsymbol{T}^{\sigma(\tau_1(x_1),\ldots,\tau_k(x_k))}\cdot X_1\cdot\ldots\cdot X_k
\end{aligned}$$

Proof. $\phi(@) \in \boldsymbol{T}^{\tau(x)}\cdot(\boldsymbol{T}^{\sigma(x_1,\ldots,x_k)}\cdot X_1\cdot\ldots\cdot X_k)$ iff $\phi(\tau(@)) \in \boldsymbol{T}^{\sigma(x_1,\ldots,x_k)}\cdot X_1\cdot\ldots\cdot X_k$. This is the case iff

$$X_1|^{x_1}_@,\ldots,X_k|^{x_k}_@ \vdash^T \phi(\tau(\sigma(x_1,\ldots,x_k))$$

which is iff $\phi(@) \in \boldsymbol{T}^{\tau(\sigma(x_1,\ldots,x_k)}\cdot X_1\cdot\ldots\cdot X_k$.

For proving the second equality assume that $\phi(@)$ is contained in the right hand side. This is iff

$$(\boldsymbol{T}^{\tau_1(x)}\cdot X_1)|^{x_1}_@,\ldots,(\boldsymbol{T}^{\tau_k(x)}\cdot X_k)|^{x_k}_@ \vdash^T \phi(\sigma(x_1,\ldots,x_k)).$$

Using the unary characterization lemma we see that this is iff

$$X_1|^{x_1}_@,\ldots,X_k|^{x_k}_@ \vdash^T \phi(\sigma(\tau_1(x_1),\ldots,\tau_k(x_k))).$$

The last step is seen immediately by remarking that

$$\psi_1(x_1),\ldots,\psi_k(x_k) \vdash^T \phi(\sigma(x_1,\ldots,x_k))$$

iff

$$\psi_1(\tau_1(x_1)),\ldots,\psi_k(\tau_k(x_k)) \vdash^T \phi(\sigma(\tau_1(x_1),\ldots,\tau_k(x_k))).$$

▮

NOTATION
To simplify notation we make the following convention for internal combinators. Let *op* be an operation in the language. Then we write

$$\boldsymbol{T}^{op(x_1,\ldots,x_k)}\cdot X_1\cdot\ldots\cdot X_k = op(X_1,\ldots,X_k).$$

Example. We write $\boldsymbol{T}^{x_1+x_2}\cdot X_1\cdot X_2 = X_1 + X_2$.

Furthermore due to the main theorem we can interchangeably use the notations $\boldsymbol{T}\cdot X_1\cdot\ldots\cdot X_k$ or $\boldsymbol{T}(X_1,\ldots,X_k)$ for denoting the application of combinatory operations to its arguments.

Fixed Point Property of Nonmonotone Recursion

We propose that the recursion combinator of the combinatory model allows two different possibilities to assign properties to recursively defined functions, by continuity and by fixed point relations. The first is realized by the definition, namely a recursion combinator has any property that is maintained throughout the recursion. We show now that also the second possibility is realized. This allows to establish algebraic relations involving recursion combinators by using fixed point equations.

We denote in the following a general nonmonotone recursion combinator $\boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\ldots,\boldsymbol{T}_k}$ simply by $\boldsymbol{M}$. Then we state the following *fixed point property.*

Proposition 8 (Fixed point property) *Let* $\boldsymbol{G}$ *be a* k*-ary combinatory operation. If* $\boldsymbol{T}^{m+1} = \boldsymbol{G}\cdot\boldsymbol{T}^{m}\cdot\ldots\cdot\boldsymbol{T}^{m-k}$ *for* $m \geq k$ *then*

$$\boldsymbol{M} = \boldsymbol{G}\cdot\boldsymbol{M}\cdot\ldots\cdot\boldsymbol{M}.$$

Proof. The proof uses a standard argument for graph models. One inclusion is trivial: The assumption of the proposition gives

$$\boldsymbol{T}^{m+1} \sqsubseteq \boldsymbol{G}\cdot\boldsymbol{T}^{m}\cdot\ldots\cdot\boldsymbol{T}^{m-k}.$$

Remember that $\boldsymbol{M} = \bigsqcup_{m\in\mathbb{N}}\boldsymbol{T}^{m}$ and so by using monotonicity of $\boldsymbol{G}$ we obtain first $\boldsymbol{T}^{m+1} \sqsubseteq \boldsymbol{G}\cdot\boldsymbol{M}\cdot\ldots\cdot\boldsymbol{M}$ and hence

$$\boldsymbol{M} \sqsubseteq \boldsymbol{G}\cdot\boldsymbol{M}\cdot\ldots\cdot\boldsymbol{M}.$$

For the other direction assume that $\phi \in \boldsymbol{G}\cdot\boldsymbol{M}\cdot\ldots\cdot\boldsymbol{M}$. This is the case iff there exist finite $\alpha_i \sqsubseteq \boldsymbol{M},\ i = 1,\ldots,k$ such that

$$\alpha_1 \to \ldots \to \alpha_k \to \phi \in \boldsymbol{G}.$$

Therefore since the α_i are finite there exist $m_1,\ldots,m_k$ such that $\alpha_i \sqsubseteq \boldsymbol{T}^{m_i},\ i = 1,\ldots,k$. Since the $\boldsymbol{T}^{m}$ are monotone increasing there exists a m_0 such that $\boldsymbol{T}^{m_i} \sqsubseteq \boldsymbol{T}^{m_0-i}$. Now we conclude that $\phi \in \boldsymbol{G}\cdot\boldsymbol{T}^{m_0}\cdot\ldots\cdot\boldsymbol{T}^{m_0-k}$ which proves that $\phi \in \boldsymbol{M}$. ▮

The following corollary justifies the use of the notion fixed point property.

Corollary 1 (Fixed point property of unary recursion) *Let* $\boldsymbol{G}$ *be a combinatory operation. Then*

$$\boldsymbol{M}^{\boldsymbol{G}}(\boldsymbol{G}^0) = \boldsymbol{G}(\boldsymbol{M}^{\boldsymbol{G}}(\boldsymbol{G}^0)).$$

Proof. The iterates $\boldsymbol{G}^m$ of the combinator $\boldsymbol{M}^{\boldsymbol{G}}(\boldsymbol{G}^0)$ satisfy $\boldsymbol{G}^{m+1} = \boldsymbol{G}(\boldsymbol{G}^m)$ and hence solve the fixed point equation. ▮

Corollary 2 *If* $\boldsymbol{G}^0 \sqsubseteq \boldsymbol{G}(\boldsymbol{G}^0)$ *then* $\boldsymbol{G}^0$ *approximates a fixed point of* $\boldsymbol{G}$.

Normal Form of Nonmonotone Recursion Combinators

In this section we will show that it is not necessary to discuss the case of composite recursion operators, since for every such combinator there exists an equivalent one, which is built up with exactly one nonmonotone recursion combinator at the outermost level and has the same extension in $\mathcal{E}_{\mathcal{A}_@}$. The idea of the proof is to eliminate recursions by diagonalization. The problem is that we have to handle the diagonalization of infinite recursions. This is possible due to the following lemma for graph algebras.

Lemma 2 *Let* $\boldsymbol{T}$ *be a combinator of a graph model and* $X_i^k, i = 1, \ldots, n,\ k \in \mathbb{N}$ *combinators such that* X_i^k *is a chain for i fixed:* $X_i^1 \sqsubseteq X_i^2 \sqsubseteq \ldots$*. Then*

$$\boldsymbol{T} \cdot \bigsqcup_{k \in \mathbb{N}} X_1^k \cdot \ldots \cdot \bigsqcup_{k \in \mathbb{N}} X_n^k = \bigsqcup_{k \in \mathbb{N}} \boldsymbol{T} \cdot X_1^k \cdot \ldots \cdot X_n^k.$$

Proof. The inclusion $\sqsupseteq$ is trivial due to monotonicity. The other inclusion is showed as follows. Let $\beta \in \boldsymbol{T} \cdot \bigsqcup_{k \in \mathbb{N}} X_1^k \cdot \ldots \cdot \bigsqcup_{k \in \mathbb{N}} X_n^k$. Then there exist $\alpha_1, \ldots, \alpha_n$, finite and $\alpha_i \sqsubseteq X_i^k$, such that $\alpha_1 \to \ldots \to \alpha_n \to \beta \in \boldsymbol{T}$. Since the α_i are finite there exist $X_{k_1}^1, \ldots, X_{k_n}^n$ such that $\alpha_i \sqsubseteq X_i^{k_i}$. Take $k_0 = \max_{i=1,\ldots,n}(k_i)$, then $\alpha_i \sqsubseteq X_i^{k_0}$ and so $\beta \in \boldsymbol{T} \cdot X_1^{k_0} \cdot \ldots \cdot X_n^{k_0}$. This shows that $\beta \in \bigsqcup_{k \in \mathbb{N}} \boldsymbol{T} \cdot X_1^k \cdot \ldots \cdot X_n^k$. ∎

We introduce a vector notation for multi-dimensional recursions, which will make the text more readable. Let the combinator $\boldsymbol{M}^{\boldsymbol{T}, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_k}(\boldsymbol{T}, \boldsymbol{T}_1^0, \ldots, \boldsymbol{T}_k^0)$ be given defining the recursion

$$\begin{aligned}
\boldsymbol{T}^{m+1} &:= \boldsymbol{T}^m \sqcup \boldsymbol{T} \cdot \boldsymbol{T}^m \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_k^m \\
\boldsymbol{T}_1^{m+1} &:= \boldsymbol{T}_1 \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_k^m \\
&\ \vdots \\
\boldsymbol{T}_k^{m+1} &:= \boldsymbol{T}_k \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_k^m
\end{aligned}$$

with the starting values $\boldsymbol{T}^0, \boldsymbol{T}_1^0, \ldots, \boldsymbol{T}_k^0$. We denote now k-tuples as vectors. Hence the combinator is written as $\boldsymbol{M}^{\boldsymbol{T}, \vec{\boldsymbol{T}}}(\boldsymbol{T}^0, \vec{\boldsymbol{T}}^0)$. Then we write for the recursion

$$\begin{aligned}
\boldsymbol{T}^{m+1} &:= \boldsymbol{T}^m \sqcup \boldsymbol{T} \cdot \boldsymbol{T}^m \cdot \vec{\boldsymbol{T}}^m \\
\vec{\boldsymbol{T}}^{m+1} &:= \vec{\boldsymbol{T}} \cdot \vec{\boldsymbol{T}}^m.
\end{aligned}$$

When working with several recursion combinators with different starting values and different arities it is always possible to find a set of equivalent recursion combinators which share the same set of starting values. This is achieved by simply introducing trivial recursions, thus raising the arity of the recursion combinator, which have no influence on the limit. For example $\boldsymbol{M}^{\boldsymbol{T}, \boldsymbol{T}_1, \ldots, \boldsymbol{T}_k}(\boldsymbol{T}, \boldsymbol{T}_1^0, \ldots, \boldsymbol{T}_k^0)$ represents the same set in $\mathcal{E}_{\mathcal{A}_@}$ as $\boldsymbol{M}^{\widetilde{\boldsymbol{T}}, \widetilde{\boldsymbol{T}}_1, \ldots, \widetilde{\boldsymbol{T}}_k, \boldsymbol{G}}(\boldsymbol{T}, \boldsymbol{T}_1^0, \ldots, \boldsymbol{T}_k^0, \boldsymbol{G}^0)$ where

$$\widetilde{\boldsymbol{T}} \cdot X \cdot X_1 \cdot \ldots \cdot X_k \cdot X_{k+1} = \boldsymbol{T}_k \cdot X \cdot X_1 \cdot \ldots \cdot X_k$$

and similarly for $\widetilde{\boldsymbol{T}}_1, \ldots, \widetilde{\boldsymbol{T}}_k$.

As a first step we show that we can pull continuous operations inside a recursion.

Lemma 3 *Let recursion combinators* $\boldsymbol{M}_i = \boldsymbol{M}_{X,\vec{X}}^{\boldsymbol{T}_i,\vec{\boldsymbol{T}}_i}(\boldsymbol{T}^0, \vec{\boldsymbol{T}}^0)$ $i = 1, \ldots, n$ *be given. Let* $\boldsymbol{T}$ *be a combinator. Then there exists a recursion combinator* $\widetilde{\boldsymbol{M}}$ *such that*

$$\boldsymbol{T} \cdot \boldsymbol{M}_1 \cdot \ldots \cdot \boldsymbol{M}_n = \widetilde{\boldsymbol{M}}.$$

Proof. Each $\boldsymbol{M}_i$ can be written as a union,

$$\boldsymbol{M}_i = \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T}_i^m.$$

Hence by the lemma we have

$$\begin{aligned} \boldsymbol{T} \cdot \boldsymbol{M}_1 \cdot \ldots \cdot \boldsymbol{M}_n &= \boldsymbol{T} \cdot \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T}_1^m \cdot \ldots \cdot \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T}_n^m = \\ &= \bigsqcup_{m \in \mathbb{N}} \boldsymbol{T} \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_n^m \end{aligned}$$

We build a recursion that gives the $\boldsymbol{T} \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_n^m$.

$$\begin{aligned} \boldsymbol{T}_1^{m+1} &:= \boldsymbol{T}_1^m \sqcup \boldsymbol{T}_1 \cdot \boldsymbol{T}_1^m \cdot \vec{\boldsymbol{T}}_1{}^m \\ \vec{\boldsymbol{T}}_1{}^{m+1} &:= \vec{\boldsymbol{T}}_1 \cdot \vec{\boldsymbol{T}}_1{}^m \\ &\vdots \\ \boldsymbol{T}_n^{m+1} &:= \boldsymbol{T}_n^m \sqcup \boldsymbol{T}_n \cdot \boldsymbol{T}_n^m \cdot \vec{\boldsymbol{T}}_n{}^m \\ \vec{\boldsymbol{T}}_n{}^{m+1} &:= \vec{\boldsymbol{T}}_n \cdot \vec{\boldsymbol{T}}_n{}^m \\ \boldsymbol{T}^{m+1} &:= \boldsymbol{T}^m \sqcup \boldsymbol{T} \cdot \boldsymbol{T}_1^m \cdot \ldots \cdot \boldsymbol{T}_n^m \end{aligned}$$

This recursion written as recursion combinator gives $\widetilde{\boldsymbol{M}}$. ∎

This lemma shows that we can pull all combinatory operations (except recursion operations) to the inside. Hence every combinatory term can be rewritten such that all recursions appear on the outside of the term. Next we show that composed recursion combinators can be diagonalized. The proof is straightforward although the notations are tedious. We again assume that all recursion combinators share the same starting values.

Lemma 4 *Let recursion combinators* $\boldsymbol{M}_i = \boldsymbol{M}^{\boldsymbol{T}_i,\vec{\boldsymbol{T}}_i}(\boldsymbol{T}^0, \vec{\boldsymbol{T}}^0)$ $i = 0, \ldots, n$ *be given. Then the term*

$$\boldsymbol{M}^{\boldsymbol{G},\boldsymbol{G}_1,\ldots,\boldsymbol{G}_n} \cdot \boldsymbol{M}_0 \cdot \boldsymbol{M}_1 \cdot \ldots \cdot \boldsymbol{M}_n$$

can be rewritten as one recursion combinator with starting values $\boldsymbol{T}^0, \vec{\boldsymbol{T}}^0$ *using the embeddings of the natural numbers and the* $\boldsymbol{Zero}$ *combinator.*

Proof. First we give a diagonalized representation of

$$\boldsymbol{M}^{\boldsymbol{G},\boldsymbol{G}_1,\ldots,\boldsymbol{G}_n} \cdot \boldsymbol{M}_0 \cdot \boldsymbol{M}_1 \cdot \ldots \cdot \boldsymbol{M}_n$$

and then a recursion which computes it.

Let $\boldsymbol{G}^l(\boldsymbol{S}_0, \boldsymbol{S}_1, \ldots, \boldsymbol{S}_n)$ denote the l-th iterate of $\boldsymbol{M}^{\boldsymbol{G},\boldsymbol{G}_1,\ldots,\boldsymbol{G}_n} \cdot \boldsymbol{S}_0 \cdot \boldsymbol{S}_1 \cdot \ldots \cdot \boldsymbol{S}_n$.

$$\begin{aligned}
&\boldsymbol{M}^{\boldsymbol{G},\boldsymbol{G}_1,\ldots,\boldsymbol{G}_n} \cdot \boldsymbol{M}_0 \cdot \boldsymbol{M}_1 \cdot \ldots \cdot \boldsymbol{M}_n = \\
&\boldsymbol{M}^{\boldsymbol{G},\boldsymbol{G}_1,\ldots,\boldsymbol{G}_n} \cdot \textstyle\bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_0^m \cdot \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_1^m \cdot \ldots \cdot \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_n^m = \\
&\textstyle\bigsqcup_{l\in\mathbb{N}} \boldsymbol{G}^l(\bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_0^m, \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_1^m, \ldots, \bigsqcup_{m\in\mathbb{N}} \boldsymbol{T}_n^m) = \\
&\textstyle\bigsqcup_{l\in\mathbb{N}} \bigsqcup_{m\in\mathbb{N}} \boldsymbol{G}^l(\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m).
\end{aligned}$$

We claim that

$$\bigsqcup_{l\in\mathbb{N}} \bigsqcup_{m\in\mathbb{N}} \boldsymbol{G}^l(\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m) = \bigsqcup_{m\in\mathbb{N}} \boldsymbol{G}^m(\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m).$$

We have only to show the inclusion $\sqsubseteq$ since the other is obvious. Let

$$\phi \in \bigsqcup_{l\in\mathbb{N}} \bigsqcup_{m\in\mathbb{N}} \boldsymbol{G}^l(\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m).$$

Then there exists l_0, m_0 such that $\phi \in \boldsymbol{G}^{l_0}(\boldsymbol{T}_0^{m_0}, \boldsymbol{T}_1^{m_0}, \ldots, \boldsymbol{T}_n^{m_0})$. Assume that $l_0 \leq m_0$. Then, since the $\boldsymbol{G}^l$ form a chain, we have

$$\boldsymbol{G}^{l_0}(\boldsymbol{T}_0^{m_0}, \boldsymbol{T}_1^{m_0}, \ldots, \boldsymbol{T}_n^{m_0}) \sqsubseteq \boldsymbol{G}^{m_0}(\boldsymbol{T}_0^{m_0}, \boldsymbol{T}_1^{m_0}, \ldots, \boldsymbol{T}_n^{m_0}).$$

Otherwise, since also all $\boldsymbol{T}_i^m$ form chains we get by monotonicity

$$\boldsymbol{G}^{l_0}(\boldsymbol{T}_0^{m_0}, \boldsymbol{T}_1^{m_0}, \ldots, \boldsymbol{T}_n^{m_0}) \sqsubseteq \boldsymbol{G}^{l_0}(\boldsymbol{T}_0^{l_0}, \boldsymbol{T}_1^{l_0}, \ldots, \boldsymbol{T}_n^{l_0}).$$

Hence the claim is proven correct.

So we have to give a recursion that computes

$$\bigsqcup_{m\in\mathbb{N}} \boldsymbol{G}^m(\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m).$$

the recursion we give is essentially built up as follows. Every time we have computed in one step the $\boldsymbol{T}_0^{m+1}, \boldsymbol{T}_1^{m+1}, \ldots, \boldsymbol{T}_n^{m+1}$ out of the $\boldsymbol{T}_0^m, \boldsymbol{T}_1^m, \ldots, \boldsymbol{T}_n^m$ we build up in $m+1$ steps $\boldsymbol{G}^{m+1}(\boldsymbol{T}_0^{m+1}, \boldsymbol{T}_1^{m+1}, \ldots, \boldsymbol{T}_n^{m+1})$. To control the execution of the recursion we make use of the embedding of natural numbers and the combinator ***Zero*** given earlier. Furthermore we need a combinator that can compute the difference of two natural numbers which can be given easily.

Now we are ready to give the recursion. We use two auxiliary combinators $\boldsymbol{C}^m, \boldsymbol{D}^m$ for counting, where $\boldsymbol{D}^m$ is the counter for the outer loop which computes the $\boldsymbol{T}_i^{m+1}$ while $\boldsymbol{C}^m$ is the counter for the inner loop computing $\boldsymbol{G}^{m+1}$. Furthermore we need auxiliary recursions for $\boldsymbol{H}^m, \boldsymbol{H}_1^m, \ldots, \boldsymbol{H}_n^m$ which hold the intermediate results for computing $\boldsymbol{G}^{m+1}$. We omit the parenthesis around the three arguments of ***Zero*** to obtain better readability.

$$\begin{array}{lllll}
\boldsymbol{C}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{C}^m + 1 & \cdot\ \underline{0} \\
\boldsymbol{D}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{D}^m & \cdot\ \boldsymbol{D}^m + 1 \\
 & & & & \\
\boldsymbol{T}_0^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{T}_0^m & \cdot\ \boldsymbol{T}_0^m \sqcup \boldsymbol{T}_0 \cdot \boldsymbol{T}_0^m \cdot \vec{\boldsymbol{T}_0}^m \\
\vec{\boldsymbol{T}_0}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \vec{\boldsymbol{T}_0}^m & \cdot\ \vec{\boldsymbol{T}}_0 \cdot \vec{\boldsymbol{T}_0}^m \\
 & \vdots & & & \\
\boldsymbol{T}_n^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{T}_n^m & \cdot\ \boldsymbol{T}_n^m \sqcup \boldsymbol{T}_n \cdot \boldsymbol{T}_n^m \cdot \vec{\boldsymbol{T}_n}^m \\
\vec{\boldsymbol{T}_n}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \vec{\boldsymbol{T}_n}^m & \cdot\ \vec{\boldsymbol{T}}_n \cdot \vec{\boldsymbol{T}_n}^m \\
 & & & & \\
\boldsymbol{H}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{H}^m \sqcup \boldsymbol{G} \cdot \boldsymbol{H}^m \cdot \boldsymbol{H}_1^m \cdot \ldots \cdot \boldsymbol{H}_n^m & \cdot\ \boldsymbol{T}_0^m \sqcup \boldsymbol{T}_0 \cdot \boldsymbol{T}_0^m \cdot \vec{\boldsymbol{T}_0}^m \\
\boldsymbol{H}_1^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{G}_1 \cdot \boldsymbol{H}_1^m \cdot \ldots \cdot \boldsymbol{H}_n^m & \cdot\ \boldsymbol{T}_1^m \sqcup \boldsymbol{T}_1 \cdot \boldsymbol{T}_1^m \cdot \vec{\boldsymbol{T}_1}^m \\
 & \vdots & & & \\
\boldsymbol{H}_n^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{G}_n \cdot \boldsymbol{H}_1^m \cdot \ldots \cdot \boldsymbol{H}_n^m & \cdot\ \boldsymbol{T}_n^m \sqcup \boldsymbol{T}_n \cdot \boldsymbol{T}_n^m \cdot \vec{\boldsymbol{T}_n}^m \\
 & & & & \\
\boldsymbol{G}^{m+1} & := & \boldsymbol{Zero} \cdot \boldsymbol{D}^m - \boldsymbol{C}^m & \cdot\ \boldsymbol{G}^m & \cdot\ \boldsymbol{G}^m \sqcup \boldsymbol{H}^m
\end{array}$$

Starting values are $\boldsymbol{C}^0 = \underline{0}$, $\boldsymbol{D}^0 = \underline{0}$, $\boldsymbol{H}^0 = \boldsymbol{T}_0^0$, $\boldsymbol{H}_i^0 = \boldsymbol{T}_i^0$ and $\boldsymbol{G}^0 = \boldsymbol{E}$. ∎

The embeddings of the natural numbers and the operations on them, which can be realized in combinatory algebras, are not basic operations in the definition of combinatory models. Therefore the diagonalization result does not hold for arbitrary combinatory models in the strict sense, that all operations used in the diagonalization are combinatory embeddings of terms or formulas. However, as far as the application to analysis is concerned, the natural numbers and the basic operations on them, like addition and subtraction, are always available as a part of the analytic structure, and hence expressible in the logic used to describe it. In these cases also a decision combinator can be constructed as an operation of the combinatory model, as will be shown later, and so the diagonalization can be realized completely within the combinatory model.

Retractions and Solutions

In this section we introduce some standard notions used in analytic computation, like rounding or approximate solution etc., by precise algebraic definitions.

Often we want to compute in $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ only with combinators of a certain type. A natural way to impose restrictions on the carrier set $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ is by retracts. Let $\boldsymbol{R}$ be a retraction. Then the carrier set of the *retracted combinatory model* is

$$\mathcal{E}_R = \{X^R = \boldsymbol{R} \cdot X,\ X \in \mathcal{E}_{A_{@_1,\ldots,@_n}}\}$$

and the retracted combinatory model is the algebraic structure

$$\mathbf{E}_R =< \mathcal{E}_R; \boldsymbol{T}^{\phi R}, \boldsymbol{M}^{\boldsymbol{T}^R, \boldsymbol{T}_1^R, \ldots, \boldsymbol{T}_k^R} >$$

where the *retracted operations* are defined as

$$\boldsymbol{T}^R \equiv \boldsymbol{B}^k \cdot \boldsymbol{R} \cdot \boldsymbol{T}.$$

If the retraction $\boldsymbol{R}$ of $\mathbf{E}_{A_{@_1,\ldots,@_n}}$ satisfies

$$\boldsymbol{R} \cdot X \sqsubseteq X, X \in \mathcal{E}_{A_{@_1,\ldots,@_n}}$$

we will call it a *proper retraction.* In a proper retraction the retract of a combinator is hence still an approximation of the retracted combinator. A special type of proper retraction can be defined by restricting the allowed formulas. Let $R \subseteq A_{@_1,\ldots,@_n}$ be the set of formulas that may be used to describe an object. Then the combinator

$$\boldsymbol{R}^- := \{\psi(@_1, \ldots, @_n) \rightarrow \psi(@_1, \ldots, @_n) : \psi(@_1, \ldots, @_n) \in R\}$$

reduces a combinator in $\mathcal{E}_{A_{@_1,\ldots,@_n}}$ to a subset of R. Since we want to consider only combinators closed under logical consequence we choose as retraction

$$\boldsymbol{R} := \boldsymbol{B} \cdot \boldsymbol{Cn} \cdot \boldsymbol{R}^-.$$

Then $\boldsymbol{R}$ satisfies $\boldsymbol{R} \cdot X \sqsubseteq X$. The combinators in $\mathcal{E}_R$ are then defined as sets of consequences of subsets of R.

Example. Assume we are computing in integer arithmetic. A retraction could then be a rounding to the next integer, for example

$$\boldsymbol{R}^{round} \cdot \{@ = 3/4\} = \{@ = 1\}.$$

A proper retraction would be to the smallest integer interval, such that the interval includes this number:

$$\boldsymbol{R}^{int} \cdot \{@ = 3/4\} = \{0 \leq @ \leq 1\}.$$

An especially interesting class of retracted operations are *complete retracted operations* which satisfy

$$\boldsymbol{T}^R \cdot X_1^R \cdot \ldots \cdot X_n^R = (\boldsymbol{T} \cdot X_1 \cdot \ldots \cdot X_n)^R.$$

Example. Assume we retract in the combinatory model of totally ordered fields to real intervals. Most of the arithmetic operations are then complete, except division when the interval contains zero. Namely

$$\boldsymbol{T}^{1/x} \cdot \{-1 \leq @ \leq 1\} = \boldsymbol{F}.$$

Now assume we use multi-interval arithmetic, which means we allow finite sets of intervals. Then also division is complete

$$\boldsymbol{T}^{1/x} \cdot \{-1 \leq @ \leq 1\} = \{@ \leq -1 \vee 1 \leq @\}.$$

The structure of combinatory models allows to distinguish under a rich number of solution types. We consider only the case of an internal combinatory model with carrier set $\mathcal{E}_{A_@}$. Assume a problem is given in form of an equation.

$$\tau_1(x) = \tau_2(x)$$

The *formal solution* of the problem is given in the combinatory model by $\boldsymbol{Cn} \cdot \{\tau_1(@) = \tau_2(@)\}$ in implicit form. The simplest type of an explicit solution is given by a term γ of the language of the underlying first order theory such that

$$\boldsymbol{T}^{\tau_1(x)} \cdot \boldsymbol{T}^{\gamma} = \boldsymbol{T}^{\tau_2(x)} \cdot \boldsymbol{T}^{\gamma}$$

This kind of solution is called *closed-form solution.* In the case, where a closed form solution does not exist, further possibilities are given. First we can weaken the information contained in a term by a retraction $\boldsymbol{R}$ with $\boldsymbol{R} \cdot X \sqsubseteq X$. Then γ is called an *approximative solution* if

$$\boldsymbol{R} \cdot (\boldsymbol{T}^{\tau_1(x)} \cdot \boldsymbol{T}^{\gamma}) = \boldsymbol{R} \cdot (\boldsymbol{T}^{\tau_2(x)} \cdot \boldsymbol{T}^{\gamma}).$$

A solution of the combinatory equation that is found in the combinatory model without referring to a term is called an *approximate solution.* It is given by a combinator $\boldsymbol{G} \in \mathcal{E}_{A_@}$ such that

$$\boldsymbol{T}^{\tau_1(x)} \cdot \boldsymbol{G} = \boldsymbol{T}^{\tau_2(x)} \cdot \boldsymbol{G}$$

or in the case of a retracted combinatory model as

$$\boldsymbol{T}_R^{\tau_1(x)} \cdot \boldsymbol{G} = \boldsymbol{T}_R^{\tau_2(x)} \cdot \boldsymbol{G}$$

Note that two approximate solutions are always possible, namely $\boldsymbol{E}$ and $\boldsymbol{F}$.

Remark. $\boldsymbol{E}$ is also very useful to handle the problem of undefined expressions in symbolic computation. Namely, the embedding of $\frac{1}{0}$ is nothing else than $\boldsymbol{E}$, since no nontrivial information is available for $\frac{1}{0}$.

$$\boldsymbol{T}^{\frac{1}{0}} = \boldsymbol{E}.$$

The weakest solution type is that of an *approximation of a solution* which is given by a combinator $\boldsymbol{G}^a \in \mathcal{E}_{A_@}$ such that for a $\boldsymbol{G}$ that satisfies $\boldsymbol{T}^{\tau_1(x)} \cdot \boldsymbol{G} = \boldsymbol{T}^{\tau_2(x)} \cdot \boldsymbol{G}$ we have

$$\boldsymbol{G}^a \sqsubseteq \boldsymbol{G}.$$

Example. Take the equation $\tau(x) = 1/2 * (x + 2/x) = x$. There exists no closed form solution but retracting to integer intervals $x = 3/2$ is an approximative solution:

$$\boldsymbol{R}^{int} \cdot \boldsymbol{T}^{3/2} = \{1 \leq @ \leq 2\}$$

and for $\tau(3/2) = 17/12$

$$\boldsymbol{R}^{int} \cdot \boldsymbol{T}^{17/12} = \{1 \leq @ \leq 2\}.$$

Consequently the combinator $\boldsymbol{G} = \{1 \leq @ \leq 2\}$ is a solution of the combinatory equation

$$X = 1/2 * (X + X/2).$$

Axiomatic Approach

Differential Fields

Differential fields are basic for the algebraic treatment of differential equations. For an introduction to the theory of differential fields see [Kaplansky, 1957]. Instead of studying extensions of differential fields by transcendental functions like the elementary extensions, as it is done in the classical study of integration and solution of differential equations, we extend differential fields by formal programs by the introduction of new function symbols, namely cond, $\circ$ and μ. The properties of these functions are given by axioms. This is in exactly the same way as an elementary extension e.g. the exponential function is introduced: in this case the new function symbol is exp and the axiom is $(\exp(f))' = \exp(f) * f'$.

We give first a proper axiomatization of differential fields in first order logic with equality. To affirm the formal point of view we choose as notation for terms that denote functions lower greek letters $\tau, \sigma, \ldots$. The language of this logic consists of the usual differential field operations $+$, $-$, $*$, $^{-1}$, $'$ and the constant symbols 0, 1, ι, where ι denotes the identity function. The axiomatization of the theory of differential fields consists of the field axioms, including the axioms for characteristic zero, [11] and the following *differentiation axioms.*

$$\begin{array}{rcl}(\sigma + \tau)' & = & \sigma' + \tau', \\ (\sigma * \tau)' & = & \sigma' * \tau + \sigma * \tau', \\ (-\sigma)' & = & -\sigma', \\ \sigma \neq 0 \rightarrow (\sigma^{-1})' & = & \sigma' * \sigma^{-2}, \\ \mathrm{const}(\tau) & \rightarrow & \tau' = 0, \\ \iota' & = & 1.\end{array}$$

The constant elements are selected by a predicate *const* satisfying the following axioms.

$$\mathrm{const}(0),\ \mathrm{const}(1),\ \mathrm{const}(\tau) \rightarrow \mathrm{const}(-\tau),$$
$$\mathrm{const}(\sigma) \wedge \mathrm{const}(\tau) \rightarrow \mathrm{const}(\sigma + \tau) \wedge \mathrm{const}(\sigma * \tau),$$
$$\mathrm{const}(\tau) \wedge \tau \neq 0 \rightarrow \mathrm{const}(\tau^{-1})$$

[11] This is a infinite set of axioms.

The subfield of constants is furthermore axiomatized as a totally ordered field using the total order predicate $<$.

Next we give axioms for conditional functions and composition. The axioms are chosen such that results about normal forms and decidability are obtainable. In this way terms built up with field operations, composition and conditionals can be understood as a natural extension of the rational terms. The axioms are a modified version of those given in [Engeler, 1990].

Composition
We give first the axioms for composition.

$$\begin{aligned}
\iota \circ \tau &= \tau \circ \iota = \tau, \\
(\tau_1 + \tau_2) \circ \sigma &= \tau_1 \circ \sigma + \tau_2 \circ \sigma, \\
((\tau_1 * \tau_2) \circ \sigma = 0 \vee \tau_1 \circ \sigma * \tau_2 \circ \sigma \neq 0) &\rightarrow (\tau_1 * \tau_2) \circ \sigma = \tau_1 \circ \sigma * \tau_2 \circ \sigma, \\
(-\tau) \circ \sigma &= -(\tau \circ \sigma), \\
\tau \circ \sigma \neq 0 \rightarrow (\tau^{-1}) \circ \sigma &= (\tau \circ \sigma)^{-1}, \\
(\tau_1 \circ \tau_2) \circ \tau_3 &= \tau_1 \circ (\tau_2 \circ \tau_3), \\
\mathrm{const}(\sigma) &\rightarrow \sigma \circ \tau = \sigma, \\
(\sigma \circ \tau)' &= \tau' * (\sigma' \circ \tau),
\end{aligned}$$

The preconditions in the axioms for multiplication and multiplicative inverse are necessary to avoid contradictions.

Example. $(\iota * \iota^{-1}) \circ 0$ and $\iota^{-1} \circ (\iota^{-1} \circ 0)$ are typical examples where such inconsistencies would occur without the preconditions.

For compositional terms consisting of the constants $0, 1$, the ring operations $+, *, -$ and the composition operation $\circ$ a *normal form algorithm* was given in [Aberer, 1990].

We can use composition to extend the definition of the total order predicate to terms denoting functions by

$$\begin{aligned}
\sigma < \tau &:\equiv \forall x \ (\mathrm{const}(x) \rightarrow \sigma \circ x < \tau \circ x), \\
\sigma \leq \tau &:\equiv \forall x \ (\mathrm{const}(x) \rightarrow \sigma \circ x \leq \tau \circ x).
\end{aligned}$$

$\leq$ has the properties of a partial order relation with respect to the equivalence relation $\simeq$ where $\simeq$ is defined as

$$\sigma \simeq \tau :\equiv \sigma \leq \tau \wedge \tau \leq \sigma.$$

Note that the relation $\simeq$ does not coincide with the equality relation $=$ when introducing conditionals.

Conditionals
The intuition behind the axiomatization of conditional functions is that a conditional function term $\mathrm{cond}\ \sigma\ \tau_1\ \tau_2$ takes at a point c all properties of τ_1 at this point if $\sigma \circ c > 0$ and otherwise those of τ_2. That means that for a term $\mathrm{cond}\ \sigma\ \tau_1\ \tau_2$ for example not only the

function value but also the derivatives at c are the same as those of τ_1 if $\sigma \circ c > 0$. This is in contrast to the usual intuition behind piecewise rational functions but this concept allows to define the differentiation of piecewise defined function terms everywhere.

To express this we have to introduce an auxiliary function symbol $\bullet$ which satisfies the following axioms.

$$
\begin{array}{rcl}
\multicolumn{3}{c}{1 \bullet c = 1,\ 0 \bullet c = 0,\ \iota \bullet c = \iota,} \\
(\sigma + \tau) \bullet c & = & \sigma \bullet c + \tau \bullet c, \\
(-\sigma) \bullet c & = & -(\sigma \bullet c), \\
(\sigma * \tau) \bullet c & = & (\sigma \bullet c) * (\tau \bullet c), \\
\sigma \bullet c \neq 0 \rightarrow \sigma^{-1} \bullet c & = & (\sigma \bullet c)^{-1}, \\
\sigma' \bullet c & = & (\sigma \bullet c)', \\
(\sigma \circ \tau) \bullet c & = & (\sigma \bullet (\tau \circ c)) \circ (\tau \bullet c), \\
\mathrm{const}(c) \wedge (\sigma \circ c > 0) & \rightarrow & (\mathrm{cond}\ \sigma\ \tau_1\ \tau_2) \bullet c = \tau_1 \bullet c, \\
\mathrm{const}(c) \wedge (\sigma \circ c \leq 0) & \rightarrow & (\mathrm{cond}\ \sigma\ \tau_1\ \tau_2) \bullet c = \tau_2 \bullet c, \\
\forall x(\mathrm{const}(x) \rightarrow \sigma \bullet x = 0) & \rightarrow & \sigma = 0, \\
\forall x(\mathrm{const}(x) \rightarrow \sigma \bullet x \neq 0) & \rightarrow & \sigma * \sigma^{-1} = 1.
\end{array}
$$

These axioms allow to derive many properties one would expect to hold for conditional functions. We list here only some of these.

$$
\begin{array}{rcl}
(\mathrm{cond}\ \tau_1\ \tau_2\ \tau_3) \circ \sigma & = & \mathrm{cond}\ \tau_1 \circ \sigma\ \tau_2 \circ \sigma\ \tau_3 \circ \sigma, \\
\sigma \circ (\mathrm{cond}\ \tau_1\ \tau_2\ \tau_3) & = & \mathrm{cond}\ \tau_1\ \sigma \circ \tau_2\ \sigma \circ \tau_3, \\
(\mathrm{cond}\ \tau_1\ \tau_2\ \tau_3) + \sigma & = & \mathrm{cond}\ \tau_1\ \tau_2 + \sigma\ \tau_3 + \sigma, \\
-(\mathrm{cond}\ \tau_1\ \tau_2\ \tau_3) & = & \mathrm{cond}\ \tau_1\ -\tau_2\ -\tau_3, \\
\tau_2 * \tau_3 \neq 0 \rightarrow (\mathrm{cond}\ \tau_1\ \tau_2\ \tau_3)^{-1} & = & \mathrm{cond}\ \tau_1\ \tau_2^{-1}\ \tau_3^{-1}, \\
(\mathrm{cond}\ \sigma\ \tau_1\ \tau_2)' & = & \mathrm{cond}\ \sigma\ \tau_1'\ \tau_2', \\
\sigma > 0 \rightarrow \mathrm{cond}\ \sigma\ \tau_1\ \tau_2 & = & \tau_1, \\
\sigma \leq 0 \rightarrow \mathrm{cond}\ \sigma\ \tau_1\ \tau_2 & = & \tau_2, \\
\mathrm{cond}\ \sigma\ \tau\ \tau & = & \tau,
\end{array}
$$

Example. The absolute value function $|\iota| = \mathrm{cond}\ \iota\ \iota\ -\iota$ is differentiable and the derivative is given by $|\iota|' = cond\ \iota\ 1\ -1$.

To see that $\simeq$ does not coincide with $=$ consider the term $\tau = \mathrm{cond}\ \iota\ 0\ (\mathrm{cond}\ -\iota\ 0\ \iota)$. It is easy to see that $\tau \simeq 0$ but $\tau \neq 0$ since $\tau' = \mathrm{cond}\ \iota\ 0\ (\mathrm{cond}\ -\iota\ 0\ 1) \neq 0$.

The term $\mathrm{cond}\ \iota\ 1\ 0$ has no multiplicative inverse but is also not equal to 0.

Due to the last example the inverse axiom for multiplication is no longer satisfied (and therefore is replaced by the last axiom contained in the axiomatization of • which is weaker) and so the theory extended by conditional terms is a differential ring, of course containing a differential field as subfield.

We denote the theory of differential fields extended with composition and conditional axioms with $CDFF$. For this theory the following decidability result was shown in [Aberer, 1991] over a real closed constant field: [12]

Theorem 3 *For a conditional term* $\tau(x_1, \ldots, x_n)$*, consisting of the constant symbols* $0, 1$ *and the function symbols* $+, -, *, \circ$ *and cond the problem*

$$\forall x_1 \ldots x_n \; \tau(x_1, \ldots, x_n) = 0$$

is decidable if the field of constants is real closed.

Recursion

To give properties of recursively defined functions, i.e. terms containing the function symbol μ, that was used in the introductory part, in an axiomatic way we have mainly two possibilities. On the one hand we can define the μ-term as the fixed point of the fixed point equation system corresponding to the recursion scheme. In certain situation this can make sense.

Example. Take the following recursion: $x_{n+1} = x_n * 1/2, x_0 = 1$. This recursion is described by the term $\mu_x^{x/2}(1) > 0$. Then the fixed point equation $x = x/2$ is satisfied by the limit of x_n, namely 0.

In the case of more complex recursions this works no longer as illustrated for the exponential function. The fixed point equation system is

$$y = y + u, \;\; u = u * \frac{\iota}{k}, \;\; k = k + 1$$

which is obviously senseless.

The other possibility is to consider those properties which are valid almost everywhere in the recursively computed sequence. For an unary μ-operator

$$\mu_x^{\tau(x)}(\tau_0)$$

this can be expressed by an *induction axiom* of the following kind.

$$(\phi(\tau^n(\tau_0)) \wedge \forall x \; (\phi(x) \rightarrow \tau(x))) \rightarrow \phi(\mu_x^{\tau(x)}(\tau_0)).$$

It was shown in [Aberer, 1991] that every structure without μ-operator can be extended to one with μ-operator, satisfying this axiom, but on the cost of introducing nonstandard elements.

[12]The theory of real closed fields can be axiomatized by an infinite set of axioms.

Example. The same example as before. The induction axiom would imply that $\mu_x^{x/2}(1) > 0$, i.e. the term represents an infinitesimal small positive number.

A extensive treatment of this approach where also many relations to nonstandard analysis can be found is given in [di Primo, 1991].

We have already seen that in combinatory models both aspects of assigning properties to recursively defined functions were realized. Hence we prefer the representation of recursion in combinatory models to the axiomatic representation.

Combinatory Differential Fields

We propose now the combinatory model of the theory $CDFF$, the differential fields theory extended by composition and conditionals, as an appropriate mathematical model for *constructive analysis.* We call this combinatory model a *combinatory differential field* and denote it by

$$\mathbf{CDF} = < \mathcal{F}_{A_{@_1,\dots,@_n}}; \boldsymbol{T}^{\phi}, \sqcap, \sqcup, \boldsymbol{M}^{\boldsymbol{T},\boldsymbol{T}_1,\dots,\boldsymbol{T}_k} > .$$

Note: Other combinatory differential fields can be constructed using other differential fields theories.

Naturally the question arises why we include the composition and conditionals of $CDFF$ to define **CDF**. On the one hand the combinatory model provides as an algebraic programming semantics all programming constructs. We have already used this to define nonmonotone recursion. Composition is provided by the $\boldsymbol{B}$ combinator and a construction for a conditional operation analogous to that of $CDFF$ can be made as follows. Given a quantifier-free formula $\phi(x)$, which represents the condition, we define a (ternary) combinator $\boldsymbol{C}^{\phi(x_1)} := \boldsymbol{T}^{\psi}$ with ψ given as the following positive existential formula

$$\psi(x_1, V_1, V_2, V_3) :\equiv (\exists x_1\ (\phi(x_1) \wedge V_1(x_1)) \wedge V_2(x_1)) \vee (\exists x_1\ (\neg\phi(x_1) \wedge V_1(x_1)) \wedge V_3(x_1)).$$

For this combinator we can show the following.

Proposition 9

$$\boldsymbol{C}^{\phi(x_1)} \cdot X_1 \cdot X_2 \cdot X_3 = \begin{cases} X_2, \ \textit{if } \phi(@_1) \in X_1, \ X_1 \neq \boldsymbol{F} \\ X_3, \ \textit{if } \neg\phi(@_1) \in X_1, \ X_1 \neq \boldsymbol{F} \\ X_2 \sqcap X_3, \ \textit{otherwise, if } \ X_1 \neq \boldsymbol{F} \\ \boldsymbol{F}, \ \textit{otherwise} \end{cases}$$

Proof. Assume $\phi(@_1) \in X_1$. Then

$$\boldsymbol{T}^{\psi} \cdot X_1 \cdot X_2 \cdot X_3 = (\boldsymbol{Pr}_1^1 \cdot (\{\phi(@_1)\} \sqcup X_1) \sqcup X_2) \sqcap (\boldsymbol{Pr}_1^1 \cdot (\{\neg\phi(@_1)\} \sqcup X_1) \sqcup X_3)$$

Since $\{\neg\phi(@_1)\} \sqcup X_1$ contains $\phi(@_1)$ and $\neg\phi(@_1)$ the second part of the intersection is equal to $\boldsymbol{F}$. For the first part we have $\{\phi(@_1)\} \sqcup X_1 = X_1$ and $X_1 \neq \boldsymbol{F}$. Hence $\boldsymbol{Pr}^1_1 \cdot (\{\phi(@_1)\} \sqcup X_1) = \boldsymbol{Pr}^1_1 \cdot X_1 = \boldsymbol{E}$. [13] Since $\boldsymbol{E} \sqcup X_2 = X_2$ and $X_2 \sqcap \boldsymbol{F} = X_2$ the assumption is proven correct in this case. For $\neg\phi(@_1) \in X_1$ the proof proceeds similarly. If X_1 contains neither $\phi(@_1)$ nor $\neg\phi(@_1)$ then both $\boldsymbol{Pr}^1_1 \cdot ((\{\phi(@_1)\} \sqcup X_1)$ and $\boldsymbol{Pr}^1_1 \cdot ((\{\neg\phi(@_1)\} \sqcup X_1)$ are equal to $\boldsymbol{E}$. In the last case where $X_1 = \boldsymbol{F}$ the result of the application obviously again is $\boldsymbol{F}$. ∎

On the other hand we have already included conditionals and composition in $CDFF$. We want to justify here why this is useful in the application of combinatory models to analysis.

When we consider composition the combinatory model provides for this the $\boldsymbol{B}$ combinator. The identity element in the combinatory model with respect to this composition is the internal combinator $\boldsymbol{T}^x$. So the following equation is satisfied for any unary combinator $\boldsymbol{G}$.

$$\boldsymbol{B} \cdot \boldsymbol{T}^x \cdot \boldsymbol{G} \cdot X = \boldsymbol{G} \cdot X = \boldsymbol{B} \cdot \boldsymbol{G} \cdot \boldsymbol{T}^x \cdot X.$$

This corresponds expressed in the language of logic to an operation $\circ_x$ with the properties

$$x \circ_x \tau = \tau = \tau \circ_x x, \ \tau(x) \circ_x \sigma = \tau(\sigma)$$

We have introduced in differential fields an identity function ι with $\iota' = 1$ and a composition $\circ$, such that ι is the identity function with respect to this composition. Furthermore the well known chain rule was given by $(\sigma \circ \tau)' = \tau' * (\sigma' \circ \tau)$. This intertwining of properties of composition and differentiation is something that cannot be realized by using the $\boldsymbol{B}$ combinator for composition. Therefore we want to use the axiomatically defined composition further. Consequently we use also the conditionals, which have many intricate relations to differentiation and composition as well, which we do not want to loose.

On the other hand the relations between the μ-operator and the operations of $CDFF$ are not as intricate and can be equally well be represented by the combinatory recursion operation.

Convergence

We have already pointed out the fact that the introduction of a formal recursion operator automatically introduces nonstandard elements. The same can be achieved by combinatory recursion. Take for example the combinator $\boldsymbol{M}^{\boldsymbol{T}^{x/2}}(\{0 < @ < 1\})$. This combinator represents an infinitesimal positive element. It solves the combinatory equation $\boldsymbol{T}^{x/2} \cdot X = X$. On the other hand also the combinator $\boldsymbol{M}^{\boldsymbol{T}^{x/2}}(\{0 \leq @ < 1\})$ is a solution of this combinatory problem. Both combinators are convergent and represent hence exact solutions in the sense of standard analysis. Obviously similar as in nonstandard analysis we

[13] The only formulas not containing $@_1$ that can be derived under $\vdash^T$ from X_1, when X_1 is consistent, are those which are valid in T.

have to introduce the concept of standard equality and identify combinators which differ only infinitesimal. Our goal in this section will be to characterize standard equality by a retraction. Then we can say in our terminology that a combinator which solves a problem up to standard equality is an approximative solution and we will consider such solutions, if they are convergent, as acceptable solutions.

We introduce some notations we will use in the following.

$$\begin{aligned} |\tau| &:\equiv \text{cond } \tau\ \tau\ -\tau, \\ \tau_{[-\lambda,\lambda]} &:\equiv \text{cond } \lambda - \iota\ (\text{cond } \iota - \lambda\ \tau\ 0)\ 0, \\ \sigma <_{[-\lambda,\lambda]} \tau &:\equiv \sigma_{[-\lambda,\lambda]} < \text{cond } \lambda - \iota\ (\text{cond } \iota - \lambda\ \tau\ 1)\ 1, \\ \sigma =_{[-\lambda,\lambda]} \tau &:\equiv \sigma_{[-\lambda,\lambda]} = \tau_{[-\lambda,\lambda]}. \end{aligned}$$

We denote positive constant terms with Qu^+ and conditional terms with Co. First we want to introduce the notions of *global convergence* and *local convergence*. A combinator $X \in \mathcal{E}_{A_@}$ is *globally convergent* if for all terms $\rho, \lambda \in Qu^+$

$$|@| <_{[-\lambda,\lambda]} \rho \in |X| = X - X.$$

This expresses that X approximates a function globally up to a arbitrarily small error. A combinator $X \in \mathcal{E}_{A_@}$ is *locally convergent* if for all terms $\rho, \lambda \in Qu^+$, there exists a term $\tau \in Co$ such that

$$|@ - \tau| <_{[-\lambda,\lambda]} \rho \in X.$$

The above formula says that the function approximated by X can locally be approximated by a conditional function with arbitrarily small error.

Lemma 5 *If $X \in \mathcal{E}_{A_@}$ is locally convergent then it is globally convergent.*

Proof. Let X be globally convergent. Then for all $\rho, \lambda \in Qu^+$ there exists a $\tau \in Co$ such that $|@ - \tau| <_{[-\lambda,\lambda]} \rho/2$. Hence

$$\begin{aligned} X|^x_@, X^y_@ &\vdash^{CDDF} |x - \tau| <_{[-\lambda,\lambda]} \rho/2, |y - \tau| <_{[-\lambda,\lambda]} \rho/2 \\ &\vdash^{CDDF} |x - y| <_{[-\lambda,\lambda]} \rho \end{aligned}$$

Hence we have $|@| < \rho \in |X|$. ∎

On the other hand the converse is not true as shown by the following example. Let

$$X = \bigsqcup_{n \in \mathbb{N}} \{@ \circ 1/n = 1 \wedge @ =_{]1/n,1/(n+1)[} 0\}$$

Then X is globally convergent since it has everywhere an uniquely defined function value but it obviously can not be locally convergent because no conditional function with a finite number of discontinuities can approximate X around 0 with error $\rho < 1$.

Note that locally convergent combinators are general enough to describe an interesting class of functions. For example all analytic functions can be described by locally convergent combinators.

We give now a lemma that will be important for the proof of the main result of this section.

Lemma 6 *Let X be a locally convergent combinator. Let $\tau \in Co$ and $\rho, \lambda, \epsilon \in Qu^+$ be given, $\epsilon < \rho$. Then at least one of the two following is true.*

$$\begin{array}{lll} X|^y_@ & \vdash^{CDDF} & \neg|y-\tau| <_{[-\lambda,\lambda]} \rho - \epsilon \\ X|^y_@ & \vdash^{CDDF} & |y-\tau| <_{[-\lambda,\lambda]} \rho + \epsilon \end{array}$$

Proof. Since X is locally convergent there exists for λ, ϵ a τ_ϵ such that $X|^y_@ \vdash^{CDDF} |y - \tau_\epsilon <_{[-\lambda,\lambda]} \epsilon$. Note that $|\tau - \tau_\epsilon| <_{[-\lambda,\lambda]} \rho$ is decidable in $CDFF$. Now if $|\tau - \tau_\epsilon| <_{[-\lambda,\lambda]} \rho$ then

$$|y-\tau| \leq_{[-\lambda,\lambda]} |y-\tau_\epsilon| + |\tau - \tau_\epsilon| <_{[-\lambda,\lambda]} \rho + \epsilon.$$

If $\neg|\tau - \tau_\epsilon| <_{[-\lambda,\lambda]} \rho$ then exists $c \in Qu$ such that $|\tau \circ c - \tau_\epsilon \circ c| \geq \rho$. Therefore

$$|y \circ c - \tau \circ c| \geq ||y \circ c - \tau_\epsilon \circ c| - |\tau \circ c - \tau_\epsilon \circ c|| \geq \rho - \epsilon.$$

This shows that $\neg|y-\tau| <_{[-\lambda,\lambda]} \rho - \epsilon$. ▮

Corollary 3 *Let Y be a locally convergent combinator and X be any combinator such that $X|^x_@ \cup Y|^y_@$ is consistent. If $X|^x_@, Y|^y_@ \vdash^{CDDF} |y-\tau| <_{[-\lambda,\lambda]} \rho$ then for all $\delta \in Qu^+$*

$$Y|^y_@ \vdash^{CDDF} |y-\tau| <_{[-\lambda,\lambda]} \rho + \delta.$$

Proof. $Y|^y_@ \vdash^{CDDF} \neg|y-\tau| <_{[-\lambda,\lambda]} \rho$ would lead to a contradiction to the consistency. Hence take $\epsilon = \delta/2$ and $\hat{\rho} = \rho + \epsilon$ and apply the previous lemma. Then we have $Y|^y_@ \vdash^{CDDF} |y-\tau| <_{[-\lambda,\lambda]} \rho + \delta$. ▮

We define *standard equality* of two combinators X and Y: $X \overset{st}{=} Y$ iff $|@| <_{[-\lambda,\lambda]} \rho \in X - Y$ for all $\rho, \lambda \in Qu^+$.

As we want to characterize standard equality by a retraction we give now the retraction that will allow us to do this. The retraction will leave only formulas of the form

$$A^{st}_@ \quad = \quad \{|@ - \tau| <_{[-\lambda,\lambda]} \rho : \lambda, \rho \in Qu^+, \ \tau \in Co\}$$

But simply restricting combinators to all formulas of this form would still allow combinators which describe nonstandard elements. Thus we have to define the retraction in a more subtle manner. Take $\boldsymbol{R}^{st}$ as follows

$$\boldsymbol{R}^{st} \quad = \quad \{\{|@ - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho} \wedge \hat{\rho} > 0\} \rightarrow |@ - \tau| <_{[-\lambda,\lambda]} \rho, \lambda, \rho \in Qu^+, \ \tau \in Co\}.$$

We have to prove first that we actually have defined a retraction.

Lemma 7 *$\boldsymbol{R}^{st}$ is a retraction.*

Proof. Since $\boldsymbol{R}^{st} \cdot X \sqsubseteq X$ we have $\boldsymbol{R}^{st} \cdot (\boldsymbol{R}^{st} \cdot X) \sqsubseteq \boldsymbol{R}^{st} \cdot X$.
To prove the other direction assume that $|@ - \tau| <_{[-\lambda,\lambda]} \rho \in \boldsymbol{R}^{st} \cdot X$. Then exists $\hat{\rho} > 0$ such that $|@ - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho} \in X$. Therefore $|@ - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho}/2 \in \boldsymbol{R}^{st} \cdot X$ and so $|@ - \tau| <_{[-\lambda,\lambda]} \rho \in \boldsymbol{R}^{st} \cdot (\boldsymbol{R}^{st} \cdot X)$. ▮

Now we are ready to formulate the main theorem of this section.

Theorem 4 *Let X, Y be locally convergent combinators such that $X|^x_@ \cup Y|^y_@$ is consistent. Then $X^{st} = Y^{st}$ iff $X \overset{st}{=} Y$.*

Proof. First we prove the simple direction. Let $X^{st} = Y^{st}$. Then for all $\rho, \lambda \in Q^+$ exists τ such that $|@ - \tau| <_{[-\lambda,\lambda]} \rho/2 \in X$. Hence $|@ - \tau| <_{[-\lambda,\lambda]} \rho/2 \in Y$. From this we get

$$X|^x_@, Y|^y_@ \vdash^{CDDF} |x - y| <_{[-\lambda,\lambda]} \rho$$

Hence we conclude $|@| <_{[-\lambda,\lambda]} \rho \in X - Y$.

To prove the other direction assume that $|@ - \tau| <_{[-\lambda,\lambda]} \rho \in X^{st}$. Then there exists $\hat{\rho}$ such that $|@ - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho} \in X$. Since $X \overset{st}{=} Y$ for every ϵ we have

$$X|^x_@, Y|^y_@ \vdash^{CDDF} |x - y| <_{[-\lambda,\lambda]} \epsilon,$$

and, from this,

$$X|^x_@, Y|^y_@ \vdash^{CDDF} |y - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho} + \epsilon.$$

Using the corollary, we get for any δ

$$Y|^y_@ \vdash^{CDDF} |y - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho} + \epsilon + \delta.$$

Setting $\epsilon, \delta = \hat{\rho}/3$, we get $|@ - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho}/3 \in Y$ and so $|@ - \tau| <_{[-\lambda,\lambda]} \rho \in Y^{st}$. ▮

Remark. The difficulty of this proof comes from the fact that we cannot conclude from $X|^x_@, Y|^y_@ \vdash^{CDDF} |y - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho}$ that $Y|^y_@ \vdash^{CDDF} |y - \tau| <_{[-\lambda,\lambda]} \rho - \hat{\rho}$. For this we had to make the foregoing preparations.

Corollary 4 *Let X, Y be locally convergent combinators such that $X|^x_@ \cup Y|^y_@$ is consistent. If $X \sqsubseteq Y$ then $X^{st} = Y^{st}$.*

Proof. Since X is locally convergent, it is also locally convergent. Hence we have $|@| <_{[-\lambda,\lambda]} \rho \in X - X$ and by monotonicity $|@| <_{[-\lambda,\lambda]} \rho \in X - Y$. The theorem then shows that $X^{st} = Y^{st}$. ▮

This shows that any additional information put into a locally convergent combinator has no effect on its equivalence class with respect to standard equality.

Combinatory Solution of Linear Differential Equations

We consider the case of a regular, linear, homogenous differential equation with polynomial coefficients which can be written in the form

$$L(y) = y^{(m)} + a_{m-1}(\iota) * y^{(m-1)} + \ldots + a_1(\iota) * y' + a_0(\iota) * y = 0.$$

It is a well known fact that the solution of such an equation is analytic in a neighborhood of any point. Furthermore the coefficients of the power-series representation of the analytic function are computable by a polynomial recursion. We will use this to construct a combinatory solution of the differential equation, in the form of a monotone increasing and recursively computed chain of approximations of the solution.

First we give the recursion for computing the power-series in a way that is especially suited for our purposes. Let $L^e(y)$ be $L(y)$ written in expanded form, i.e. the derivatives $y^{(i)}$ are distributed over the polynomials $a_i(\iota)$. [14] Then substitute every monomial of the form

$$c * \iota^k * y^{(i)} \quad \text{by} \quad c * \iota^k * y^{(i)}_{n+(i-m)-k},$$

where c is the constant coefficient in the polynomial. The y_n will turn out to be the power-series of the solution truncated at the n-th power. This substitution gives an operator

$$\widetilde{L}(y_n, \ldots, y_{n-t}),$$

where

$$t = \max_{i=1,\ldots,m-1} deg(a_i) - (i - m).$$

We make the ansatz

$$y_{n+1} := y_n + u_n, \; u_n = b_n * \iota^n, \; b_n \text{ constant}$$

for the recursion and assume that

$$\widetilde{L}(y_n, \ldots, y_{n-t}) = 0.$$

Note that $\widetilde{L}(y, \ldots, y) = L(y) = L^e(y)$ and

$$\widetilde{L}(y, \ldots, y) \equiv L^e(y).$$

If we make the substitution of our ansatz and since $\widetilde{L}$ is linear in y_i this assumption is satisfied iff

$$\widetilde{L}(u_n, \ldots, u_{n-t}) = 0.$$

This implies that the u_n satisfy a condition of the form

$$u_n = \sum_{i=0}^{t-1} p_i(n) * \iota^i * u_{n-i-1},$$

[14] This means that $L(y) = L^e(y)$ but $L(y) \not\equiv L^e(y)$, where $\equiv$ denotes syntactic equivalence. This will be of importance in the combinatory embedding later.

where $p_i(n)$ are polynomials in n. This gives the desired recursive computation of y_n. The starting values of the recursion have to be determined out of the initial values of the differential equations.

Up to now we have only reformulated well known facts. The crucial step now is to make a combinatory ansatz. We want to recursively compute a increasing chain of approximations of the form

$$X_n := y_{n-t} + W_1(n) * u_{n-1} + \ldots + W_t(n) * u_{n-t} = y_{n-t} + \sum_{i=1}^{t} W_i(n) * u_{n-i}.$$

We substitute this ansatz into $\widetilde{L}$ and then use the following combinatory laws which follow from the properties given for internal combinators:

$$(X+Y)' = X' + Y',\ (C * X)' = C * X',\ C * X + C * Y \sqsubseteq C * (X+Y).$$

Using linearity in $\widetilde{L}$ we get

$$\begin{aligned}\widetilde{L}(X_n, \ldots, X_{n-t}) &= \widetilde{L}(y_{n-t}, \ldots, y_{n-2t}) + \widetilde{L}(W_1(n) * u_{n-1}, \ldots, W_1(n) * u_{n-t-1}) + \ldots \\ &\quad + \widetilde{L}(W_t(n) * u_{n-t}, \ldots, W_t(n) * u_{n-2t}) \\ &\sqsubseteq W_1(n) * \widetilde{L}(u_{n-1}, \ldots, u_{n-t}) + \ldots W_t(n) * \widetilde{L}(u_{n-t-1}, \ldots, u_{n-2t}) = 0.\end{aligned}$$

If we assume that the X_n are a monotone increasing chain then we may conclude the following using continuity.

$$\bigsqcup_{n \in \mathbb{N}} \widetilde{L}(X_n, \ldots, X_{n-t}) = \widetilde{L}(\bigsqcup_{n \in \mathbb{N}} X_n, \ldots, \bigsqcup_{n \in \mathbb{N}} X_n) = L^e(\bigsqcup_{n \in \mathbb{N}} X_n) \sqsubseteq 0.$$

Observe how we made use of the syntactical equivalence of $\widetilde{L}(y, \ldots, y)$ and $L^e(y)$, such that no weakening occurred in this step. This result means that if $L^e(\bigsqcup_{n \in \mathbb{N}} X_n)$ is strong convergent then $\bigsqcup_{n \in \mathbb{N}} X_n$ which is obviously expressible as a combinatory term, is an approximative solution of $L^e(y) = 0$ in a very strong sense, namely with respect to standard retraction.

Now we want to investigate under which circumstances the sequence X_n is a chain. This will also make clear why the ansatz for X_n was chosen exactly this way. To do this we make the assumption that we are only interested in the solution on a certain interval $\boldsymbol{C} = \{a \leq @ \leq b\}$, where $-1 \leq a \leq b \leq 1$. This will allow us to use the inclusion $\boldsymbol{C} \sqsubseteq \iota^k$. Then we get

$$\begin{aligned}X_{n+1} &= y_{n-t+1} + \sum_{i=0}^{t-1} W_{i+1}(n) * u_{n-i} \\ &= y_{n-t} + u_{n-t} + \sum_{i=0}^{t-1} W_{i+1}(n) * u_{n-i} \\ &= y_{n-t} + u_{n-t} + W_1(n) * \sum_{i=1}^{t} p_i(n) * u_{n-i} * \iota^i + \sum_{i=1}^{t-1} W_{i+1}(n) * u_{n-i}\end{aligned}$$

$$\begin{aligned} &= y_{n-t} + u_{n-t} * (1 + W_1(n) * p_t(n) * \iota^t) + \sum_{i=1}^{t-1} u_{n-i} * (W_{i+1}(n) + W_1(n) * p_i(n) * \iota^i) \\ &\sqsupseteq y_{n-t} + u_{n-t} * (1 + W_1(n) * p_t(n) * \boldsymbol{C}) + \sum_{i=1}^{t-1} u_{n-i} * (W_{i+1}(n) + W_1(n) * p_i(n) * \boldsymbol{C}) \end{aligned}$$

The X_n form a chain if the last line is an approximation of

$$X_n = y_{n-t} + W_1(n) * u_{n-1} + \ldots + W_t(n) * u_{n-t} = y_{n-t} + \sum_{i=1}^{t} W_i(n) * u_{n-i}.$$

This is the case if the following system of inclusions is satisfied.

$$\begin{aligned} W_t(n-1) &\sqsubseteq 1 + W_1(n) * p_t(n) * \boldsymbol{C} \\ W_i(n-1) &\sqsubseteq W_{i+1}(n) + W_1(n) * p_i(n) * \boldsymbol{C}, \; i = 1, \ldots, t-1 \end{aligned}$$

These inclusions are satisfied if the corresponding equations are satisfied. The corresponding system of equations looks much like a linear system of equation. In fact if assuming that $\boldsymbol{C}$ is a real number and the W_i are independent of n we have a linear system of t equations in t unknowns, which (in general) [15] is nondegenerate. We leave it as an open question whether the combinatory system is solvable in general, but it seems likely due to the arguments given.

We illustrate the algorithm for a concrete example which was computed by using *mathematica*. Take the differential equation

$$y''' + y' + x * y = 0.$$

The recursion is computed as

```
               4                     2                  2
               j  u[-4 + n]       2 j  u[-2 + n]     j  n u[-2 + n]
 u[n] -> -(---------------) + --------------- - ---------------
                       2    3                2    3                2    3
              2 n - 3 n   + n     2 n - 3 n   + n    2 n - 3 n   + n
```

The equation to be satisfied are

```
{V[2, n] == V[1, -1 + n],

     Int[-1, 0] V[1, n]
>    ------------------ + V[3, n] == V[2, -1 + n], V[4, n] == V[3, -1 + n],
         (-1 + n) n
```

[15]It is easy to compute in this case the determinant and to see that only for exceptional values of $\boldsymbol{C}$ and n the system degenerates.

```
          Int[-1, 0] V[1, n]
>     1 + ------------------ == V[4, -1 + n]}
          (-2 + n) (-1 + n) n
```

One solution of this system is

```
V[1,n_]:=Int[0,1];
V[2,n_]:=Int[0,1];
V[3,n_]:=Int[1-1/(n+2)/(n+1)/n,1];
V[4,n_]:=Int[1-1/(n+1)/(n-1)/n,1];
```

The first few iterates are then as follows.

```
          3       1         2                 5            23
X[3]   = J  Int[-(-), 0] + J  Int[0, 1] + Int[-, 1] + J Int[--, 1]
                  6                           6            24

              3       1         4       1              23        2     59
X[4]   = 1 + J  Int[-(-), 0] + J  Int[-(-), 0] + J Int[--, 1] + J  Int[--, 1]
                      6                 8              24              60

                  3       1     119      4       1         5        1
X[5]   = 1 + J + J  Int[-(-), -(---)] + J  Int[-(-), 0] + J  Int[-(---), 0] +
                          6     720              8                 120

                  2    3       1     119      4       1     209
X[6]   = 1 + J + J  + J  Int[-(-), -(---)] + J  Int[-(-), -(----)] +
                               6     720              8     1680

      5        1          6        1
>    J  Int[-(---), 0] + J  Int[-(---), 0]
              120                240

                       3
                  2    J     4       1     209       5        1       67
X[7]   = 1 + J + J  - -- + J  Int[-(-), -(----)] + J  Int[-(---), -(----)] +
                       6             8     1680              120     8064

      6        1          7          1
>    J  Int[-(---), 0] + J  Int[0, ----]
              240                 1008
```

Note that the iterates are monotone increasing combinators.

Conclusion

The goal of this work was naturally not a final consideration of the theory of combinatory differential fields. On the contrary, we wanted to give a starting point to a new way to view algorithmic analysis in an algebraic framework. So many questions and directions for further investigations are left open. Some of these we have already considered in this work, some ideas can be found in [Aberer, 1991], others are left out completely. We want to give some hints how the theory of combinatory differential fields could be continued.

The main motivation to introduce algorithmic extensions is surely the intention to incorporate approximate methods, which are usually subject of numeric analysis, for the (symbolic) solution of equations. We have illustrated this with two examples. In the future one central interest are ordinary and partial differential equations. A first step was given in this work. Other methods have to be discussed for boundary value problems and then partial differential equations. A rich source of experience in this direction can be found in the recent developments of interval methods and methods of verified inclusions [Weissinger, 1988].

As the examples have shown, combinatory differential fields are not only useful for theoretic discussions but give a good starting point for implementations. In the course of implementing combinatory algorithms, as we have seen, suddenly questions like parallelization and randomization of algorithms appear, depending on the kind of knowledge representation. It would also be most wishful to develop a computer algebra system based on the principles of combinatory models. A way how this could be realized in the computer algebra language *Scratchpad* [Jenks *et al.*, 1988] was sketched in [Aberer, 1991]. The use of this abstract data type language allows to implement generic algorithms independently of the knowledge representation and to use them afterwards in different retracted combinatory differential fields.

In [Blum *et al.*, 1989] the concept of a *recursively enumerable set* of real numbers was introduced. It is easy to see that a set recursively enumerable in this sense is exactly the complement of a set that can be described by a combinatory term and hence possesses a *recursively enumerable approximation.* It was also shown in [Aberer, 1991] that functions computable in the sense of *recursive analysis* can be approximated by combinators up to standard equality. So the question is which sets (of functions) can be approximated by combinators and which not. Of course this depends on several choices on which base set of formulas and operations one allows.

For discussing *complexity theory* there are two approaches which can give the essential ideas. In *information-based complexity theory* [Traub *et al.*, 1988] the notion of *asymptotic complexity* was defined. In [Weihrauch, 1980] an approach to extend *axiomatic complexity theory* to CPO's was given. Both ideas could be represented by discussing *complexity functions* of the form

$$\kappa(\boldsymbol{P}, X, \varepsilon)$$

where $\boldsymbol{P}$ is a combinator representing a program, X is the input and ε is the accuracy with respect to some weight that has to be defined. Then κ gives the number of steps to compute the result up to this accuracy for the given input. Again the theory depends which choices are made for admissible approximations and operations on them.

It is also thinkable to formulate results in the sense of information-based complexity, where principal limitations on possible solutions are given depending on the information contained in the input. Such results are depending much less on choices of admissible approximations and operations.

References

[Aberer, 1989] Aberer, K. (1989). Normal Forms in Combinatory Differential Fields. *ETH-Report No.* **89-01**.

[Aberer, 1990] Aberer, K. (1990). Normal Forms in Function Fields. *Proceedings ISSAC '90*, 1-7.

[Aberer, 1991] Aberer, K. (1991): Combinatory Differential Fields and Constructive Analysis. *ETH-Thesis,* **9357**.

[Blum *et al.*, 1989] Blum, L., Shub, M., Smale, S. (1989). On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness. Recursive Functions and Universal Machines, *Bulletin of AMS, Vol.* **21**, *No. 1.*

[Buchberger *et al.*, 1983] Buchberger, B., Collins, B., Loos, R. (1983).
Computer Algebra-Symbolic and Algebraic Computation. *Springer, Wien, New York.*

[Davenport *et al.*, 1988] Davenport, J.H., Siret, Y. and Tournier, (1988). Computer Algebra. *Academic Press, N.Y..*

[Dekker, 1969] Dekker, T.J., (1969). Finding a Zero by Means of Successive Linear Interpolation. *Constructive Aspects of the Fundamental Theorem of Algebra, ed. B. Dejon, P. Henrici.*

[di Primo, 1991] di Primo, B., (1991). Nichtstandard Erweiterungen von Differentialkörpern. *ETH Thesis.*

[Engeler, 1981A] Engeler, E., (1981). Metamathematik der Elementarmathematik. *Springer Verlag.*

[Engeler, 1981B] Engeler, E. (1981). Algebras and Combinators. *Algebra Universalis*, 389-392.

[Engeler, 1984] Engeler, E. (1984). Equations in Combinatory Algebras. *Proceedings of "Logic of Programs '83", SLNCS* **164**.

[Engeler, 1988] Engeler, E. (1988). A Combinatory Representation of Varieties and Universal Classes. *Algebra Universalis,* **24**.

[Engeler, 1990] Engeler, E. (1990). Combinatory Differential Fields. *Theoretical Computer Science* **72**, 119-131.

[Fehlmann, 1981] Fehlmann, T. (1981). Theorie und Anwendung des Graphmodells der kombinatorischen Logik. *Berichte des Instituts für Informatik der ETH* **41**.

[Kaplansky, 1957] Kaplansky, I. (1957). An Introduction to Differential Algebra. *Paris: Hermann.*

[Kaucher, 1983] Kaucher, E. (1983): Solving Function Space Problems with Guaranteed Close Bounds. *In Kulisch, U. and Miranker, W.L.: A New Approach to Scientific Computation , Academic Press, New York, p 139-164.*

[Krawczyk, 1983] Krawczyk, R. (1983). Intervalliterationsverfahren. *Freiburger Intervall Berichte* **83/6**.

[Jenks *et al.*, 1988] Jenks, R.D., Sutor, S.S., Watt, M.W., (1988). Scratchpad II: An Abstract Datatype System for Mathematical Computation *Mathematical Aspects of Scientific Software, ed. Rice, J.R., The IMA Volumes in Mathematics and its Applications, Springer.*

[Jensen, 1972] Jensen, A. (1972). A Computer Oriented Version of Nonstandard Analysis *in: Contributions to Non-Standard Analysis, Luxemburg, W.A.J., Robinson, A. , ed., Noth-Holland, Amsterdam.*

[Linz, 1988] Linz, P. (1988). A Critique of Numerical Analysis. *Bulletin AMS, Vol. 19, No..*

[Maeder, 1986] Mäder, R. (1986). Graph Algebras, Algebraic and Denotational Semantics. *ETH-Report* **86-04**.

[Pour-El & Richards, 1989] Pou8-El, E., Richards, I.J. (1989). Computability in Analysis and Physics. *Springer.*

[Richter & Szaeo, 1983] Richter, M.M., Szaeo, M.E. (1983). Nonstandard Computation Theory. *Colloquia Mathematica Societatis Janos Bolyai 42.*

[Seeland, 1978] Seeland, H. (1988). Algorithmische Theorien und konstruktive Geometrie. *HochschulVerlag, Stuttgart, Hochschulsammlung Naturwissenschaft, Informatik, Band 2.*

[Traub *et al.*, 1988] Traub, J.F., Wasilkowski G.W., Wozniakowski H., (1988). Information Based Complexity. *Academic Press, New York.*

[Weibel, 1990] Weibel, T. (1990). Extension of Combinatory Logic to a Theory of Combinatory Representation. *Technical Report ETH Zürich, Mathematik.*

[Weihrauch, 1980] Weihrauch, K. (1980). Rekursionstheorie und Komplexitätstheorie auf effektiven CPO-S. *Informatikberichte FernUniversität Hagen, 9/1980.*

[Weissinger, 1988] Weissinger, J. (1988). A Kind of Difference Methods for Enclosing Solutions of Ordinary Linear Boundary Value Problems. *Computing, Suppl. 6*, 23-32.

[Wolfram, 1988] Wolfram, S. (1988). Mathematica. *Addison-Wesley Publishing Company.*