Continuity, proof systems and the theory of transfinite computations

Dag Normann*

September 12, 1996

1 Introduction

1.1 Aims and motivation

The purpose of this paper is to show how the concept of transfinite computations relative to certain functionals of type 3 can be used to construct topologies and transfinite proof systems adding extra structure to some sets transfinitly definable over the continuum. Our aim is to initiate a fine-structure analysis of sets of the form $L_\kappa(HC)$. We will motivate this below.

Before doing so, let us make a few remarks on the choice of terminology. Accepting to a large extent the argumentation of Soare [42] and realizing that the raw material of most of the constructions of the paper are transfinite analogues of computations, we will use the expression 'theory of transfinite computations' to cover at least the part of 'higher recursion theory' that deals with generalisations of computations. We will also use the expressions 'computable' and 'computable relative to'. When we use the word 'recursion' or any of its derivatives it will be in a context where the use has a well-known technical interpretation, like in 'recursivly inaccessible ordinal', or in the case where we use the recursion theorem or the fixpoint theorem for domains.

The investigation of the natural numbers and of sets of natural numbers is an important part of mathematics. Computability theory has played its rôle

*The research for this paper is partly supported by the EU Science Plan, contract no. SCI*CT91-724

in this investigation. Concepts like *hyperarithmetic*, ${}^2E$, *recursivly inaccessible*, *positive induction*, *recursivly Mahlo* and *superjump* all reflect methods of defining new subsets of $\mathbb{N}$ from below, and the investigation of these concepts gives us basic information about subsets of $\mathbb{N}$.

In particular, the computable fragment of ω-logic lives well within $L_{\omega_1^{CK}}$. The proof-theoretical application of ω-logic for analysing the strength of Peano Arithmetic is well established.

In recent years, proof-theoretical connections between type-theories and fragments of set theory are established, see e.g. Rathjen [37], Griffor and Rathjen [17] or Setzer [41] . These results show a deep connection between the proof-theoretical strength of various closure principles in type theory and in computability theory.

The set of natural numbers is one important mathematical structure, the continuum is another. The ultimate aim is to develope a set of tools for investigating the continuum analogue to the above mentioned tools for investigating the natural numbers. Thus, in a sense, we will investigate superstructures for 2'nd order number theory.

We will use the topology inherited from domain representations to ad some extra structure to the sets in $L_\kappa(HC)$. We will survey the method of representation below. The new construction in this paper is the developement of a class of transfinite logics generalising ω-logic. The construction is inspired from the β-logic of Girard, see [14].

We propose to use the set HC of hereditarily countable sets as our version of the continuum. In a sense this is the most general version, as the set HF of hereditarily finite sets is the most general datastructure.

1.2 Basic concepts

1.2.1 The theory of transfinite computations

The theory of transfinite computations was initiated by Kleene [20]. One of his motivations for extending computability theory to the full hierarchy of functionals of finite, pure types was to have a tool for investigating the strength of quantification over infinite sets.

The hierarchy of types is defined as follows

$Tp(0) = \mathbb{N}$

$Tp(k+1)$ = the set of total functions mapping $Tp(k)$ into $\mathbb{N}$.

In an inductive definition with nine clauses (S1 - S9) Kleene defines the relation

$$\{e\}(\vec{\phi}) \approx n$$

where e and n are natural numbers and $\vec{\phi}$ is a finite sequence of functionals. The expression is to be read as

Algorithm no. e with input $\vec{\phi}$ terminates and gives output n

The nine clauses, or *schemes* as they are usually called in computability theory, represent basic operations on $\mathbb{N}$ (S1 - S3), composition (S4), primitive recursion on $\mathbb{N}$ (S5), permutation of input arguments (S6), oracle call of type 1 (S7), higher type oracle call (S8) and diagonalisation (S9).
S8 has the format

$$\{e\}(\vec{\phi}) \approx \phi_1(\lambda\psi\{e_1\}(\psi, \vec{\phi}))$$

where termination will require that $\{e_1\}(\psi, \vec{\phi})$ terminates for all ψ of the appropriate type.
S9 has the format

$$\{e\}(d, \vec{\phi}, \vec{\psi}) \approx \{d\}(\vec{\phi}).$$

All indices e are chosen such that they code the types of the acceptable inputs etc. For further details we refer to the original paper Kleene [20], or to the following items in our reference list [19, 28, 33, 34, 39].

Of special interest to us are computations relative to the type 3 functional ${}^3E$ defined by

$${}^3E(F) = 0 \text{ if } F(f) = 0 \text{ for all } f \text{ of type 1}$$

$${}^3E(F) = 1 \text{ if } F(f) \neq 0 \text{ for some } f \text{ of type 1.}$$

Computability in ${}^3E$ represent a transfinite extension of second order definability over $\mathbb{N}$. This view is quite analogue to the view that the hyperarithmetical sets represent a transfinite extension of first order definable sets.
In section 1.3.1 we will survey some of the known results about computations relative to ${}^3E$.

Using computability is one way of extending the analytic hierarchy transfinitly, using positive induction is another. If we consider subsets of $Tp(1)$

defined by positive induction we will get more complex sets. In the paper Barwise, Gandy & Moschovakis [2] the relationship between positive induction and admissible structures is established. It is shown that the closure ordinal of a positive induction over $Tp(1)$ may be the first ordinal where $L_\kappa(Tp(1))$ is Σ_1-admissible, or in other words, is a model of Kripke-Platek set theory. See Hinman [19] for further information. We see the theory of admissible structures as a part of the theory of transfinite computations.

Normann [30] adopted Kleene's approach and defined a notion of computations where both inputs and outputs could be arbitrary sets. This computation theory is called *Set Recursion* or *E-recursion*. Restricting the set of possible inputs gives us various interesting subtheories, Sacks [39] gives a detailed introduction. Set recursion accepting HC and its elements as inputs will be equivalent to the computation theory of ${}^3E$. Working with sets of the form $L_\kappa(HC)$ it will sometimes be easier to work with set recursion than with computations relative to ${}^3E$, because we need less coding.

In general, what is offered us from the theory of transfinite computations is a set of precise notions of complexity based on various principles for extending quantification over $Tp(1)$ through various transfinite levels. Thus the theory of transfinite computations is at least an important conceptual tool in describing what we mean by structures $L_\kappa(HC)$ where κ can be reached from below. We will later see that this theory also offer results relevant for the investigation of such structures.

1.2.2 The language of type theory

A type theory as defined by Martin-Löf [27] will be a formal theory where the basic ingredients are language and proofs. We will not be concerned with type theory as a formal theory in this paper. We will utilise the expressive power of type theory to define domains and elements in domains, see sections 1.2.3 and 1.2.4. In this section we will describe the kind of language we will borrow from type theory and the intuitive interpretation. We will be more precise in the sections on domains.

One important concept is that of a dependent family $\{B_x\}_{x\in A}$ which to us just will be an indexed family of sets where B_x in some nice way depends on $x \in A$. The standard situation will be that B_x is defined explicitly from $x \in A$, or via the fix point solution of some recursive operator.

There are several ways to define new structures from a dependent family,

the three most basic are *dependent sum*, *dependent product* and the W-*type*. We describe these:

Dependent sum If $\{B_x\}_{x\in A}$ is a dependent family, we define the sum $\sum(x \in A)B_x$ as the set of ordered pairs (x, y) where $x \in A$ and $y \in B_x$.

Dependent product If $\{B_x\}_{x\in A}$ is a dependent family we define the product $\prod(x \in A)B_x$ as the set of nice (continuous, total or whatever we might mean by nice in the particular context) functions f defined on A and with $f(x) \in B_x$ for all $x \in A$.

W-type If $\{B_x\}_{x\in A}$ is a dependent family we define $W(A, B)$ by induction as follows: If $x \in A$ and $f : B_x \to W(A, B)$ then $(x, f) \in W(A, B)$.
The induction will start with $(x, f) \in W(A, B)$ if B_x is empty and f is the empty function. We will return to this construction in section 2.4.

One important idea in type theory is the use of *universes*. A universe is a type whose elements are in turn types. In the formal theory one presumes in an axiomatic way that a universe has certain elements and is closed under certain operations. When we use universes, they will be semantical entities obtained by closing some set of objects under some set of operations. The use of universes will be a parallell to closing off under some notion of transfinite computations. Thus we see a universe operator as an analoge of a jump operator.

One group of types that is central in type theory but that is not important to us will be the types for equality. In type theory, every statement has to be identified with a type, and an element of the type will represent a verification of the statement. We will operate with a semantical notion of equality, so our concern will not be wether equality is provable, but for which special cases it is decidable. Thus Eq-types, I-types etc. play no rôle in our use of the type theory language.

We will not use any semantical equivalents to second order type constructs like e.g. in System F, see Girard [13, 16]. Since we are interested in structures that can be reached from below, second order type theory offers no natural constructors useful to our project.

1.2.3 Domains

Domain theory will be one important tool when we link concepts from the theory of transfinite computations to concepts inherited from type theory. The study of domains originate from Scott [40], but Ershov, (see e.g. [8, 9, 10]) independently gave contributions which must be seen as foundational for domain theory. Stoltenberg-Hansen & al. [44] is a sufficient introduction to the theory of domains for our purpose, and our basic definitions are taken from there. We will, however, give a very brief introduction to domain theory here.

We see domain theory as a method of studying infinite consistent sets of information via finite approximations. A domain $(D, \leq)$ will be a complete, partial ordering of a special sort. A partial ordering is complete when two properties are satisfied:

i) Every bounded, finite subset of D has a least upper bound. (In particular, the empty set has a least upper bound, i.e. D has a least element normally denoted $\bot$.)

ii) Every directed subset of D has a least upper bound.

An element x_0 of D is called *compact* if whenever x_0 is bounded by the least upper bound of a directed set, then x_0 is bounded by one of the elements of the directed set. It is easy to see that $\bot$ will be compact and that the least upper bound of a finite, bounded set of compacts is itself compact.
A complete partial ordering is called an (algebraic) *domain* if every $x \in D$ is the least upper bound of the set of compacts bounded by x.

The compacts will represent finite approximations to information, and we will say that two compacts are *consistent* when they are bounded. In this respect we can say that domains model a situation where sets of information are approximated by sets of finitary information.

We will not need a detailed knowledge of domain theory, and we will now give a quick summary of what is needed. For further details see Stoltenberg-Hansen & al.[44] or e.g. Abramsky and Jung [1].

Flat domains: A flat domain will be a domain where all elements exept $\bot$ are maximal, e.g. $\mathbb{N}_\bot = \{\bot, 0, 1, 2, 3, 4, \ldots\}$ with $\bot < n$ for all natural numbers n. We may occationally refer to the boolean values as a flat domain.

Finite sums and products: If D_1 and D_2 are two domains we may form the product $D_1 \times D_2$ as the cartesian product of the two ordered sets.
Moreover we may form the disjoint union of the two orderings. Then there is a choice between identifying the two $\perp$-elements and adding a new $\perp$ underneath the disjount union. The former approach will be consistent with our generalisation to dependent sums.

Function spaces: If D_1 and D_2 are two domains and $F : D_1 \to D_2$, we call F *continuous* if F is order preserving and preserves the least upper bound of directed sets.
If F and G are continuous, we let $F \sqsubseteq G$ if $F(x) \leq G(x)$ for all $x \in D_1$.
The set of continuous functions with this ordering is again a domain. We will not need the details of this proof here.

Dependent families: Palmgren and Stoltenberg-Hansen [36] defined the notion of continuously parameterised families of domains observing that a domain can be seen as a category with one unique morphism from x to y just in the case when $x \leq y$. The class of domains will also be a category using the so called *projection pairs* as morphisms, and a parameterisation will then be a functor F with some continuity properties. We may use the notation (D, F) or $\{E_d\}_{d \in D}$ for parameterisations.
We will not require the full technical definition of parameterisations for the details carried out in this paper. We just observe that the definition of a parameterisation $\{E_d\}_{d \in D}$ involves a chosen morphism between E_{d_1} and E_{d_2} when $d_1 \leq d_2$.

$\sum$-constructions: If $\{E_d\}_{d \in D}$ is a parameterisation of domains, if $d \in D$, $d' \in D$ with $d \leq d'$ and if $e \in E_d$, we let $e^{d'}$ denote the element of $E_{d'}$ obtained by applying the morphism from E_d to $E_{d'}$ to e. We then organise the dependent sum $\sum(d \in D)E_d$ to a domain by letting $(d, e) \leq (d', e')$ if and only if $d \leq d'$ and $e^{d'} \leq e'$. The fact that this is a domain is proved in detail in [36].

$\prod$-constructions: If $\{E_d\}_{d \in D}$ is a parameterisation of domains, we define the dependent product $\prod(d \in D)E_d$ as the set of choice functions that will

commute with direct limits when composed with the morphisms of the parameterisation. We will use the pointwise ordering to organise this set as a domain. The details can again be found in [36].

Fix points: One simple, but important result about complete partial orderings, and thus about domains, is the fix point theorem. In its simple form it just states that if $f : D \to D$ is continuous, then there is a unique least element $d \in D$ with $f(d) = d$. d will just be the limit of all $f^n(\perp)$, which will be an increasing sequence. In applications of this we will construct domains, parameterised families of domains and continuous operators as minimal solutons to equations of the appropriate kinds. This is basic domain theory, and we will not go into further details when we apply the fix point theorem.

1.2.4 Domains with totality and density

Domains are themselves simple structures. We will obtain the complex structures when we, in most cases in a canonical way, declare some of the elements of a domain to be total. In the case of the flat domain $\mathbb{N}_\perp$ all natural numbers will be total, representing complete information, while the element $\perp$ will not be total. In the case of a function space $D_1 \to D_2$ a continuous function F will be total if $F(d_1)$ is total in D_2 whenever d_1 is total in D_1. In the case of a cartesian product, a pair is total if both coordinates are total in the respective domains.
In the case of a parameterisation $\{Y_x\}_{x \in X}$ we will require that Y_x has a distinguished set of total objects whenever x is total in X. It is then trivial to extend the definition of totality to dependent sums and dependent products.

Though the total objects in most cases will be canonically given, an investigation of domains with totality will require an abstract analysis of the concept of totality. A systematic investigation of this sort was initiated independently by Kristiansen and Normann (see [23, 24, 31]) at one hand and Berger [3, 4] at the other. Berger in particular focused on the notion of totality for domains.

If X is a domain with a set $\bar{X}$ of total objects, we say that $\bar{X}$ is dense in X if every compact in X can be extended to an element of $\bar{X}$. Berger was mainly interested in cases where density is effectivly preserved, and his concept of totality reflects this. In order to preserve density through the

function space construction one has to assume a dual property which Berger [3, 4] call totality, but which we call co-density:

Definition 1 We let $\mathbb{B}$ be the flat domain of boolean values.
A subset $\bar{X}$ of a domain X is *co-dense* if for every unbounded, finite subset A of X there is a total map $f : X \to \mathbb{B}$ that in addition is total on A, but not constant on A.

Berger [4] shows that if two domains with totality satisfies both density and co-density, then so will the function space do. In Normann [32] and in Kristiansen and Normann [24] new examples of domains with totality, with density and co-density was constructed via a systematic extension of the domain interpretations of the finite types to transfinite types. Berger [5] analysed these ad hoc constructions and extended the notions of density and co-density to parameterisations, proving a general density-co-density-theorem covering all known examples. For further applications of this, see Normann [34] or the survey in section 1.3.2.

There is one consequence of density that is vital to our applications.

Definition 2 A domain X is *separable* if the set of compacts is countable.

All constructions discussed so far will preserve separability.

If X is a separable domain, $\bar{X}$ a subset of X satisfying co-density, let $\{A_n\}_{n\in\mathbb{N}}$ be an enumeration of all unbounded, finite sets of compacts. Let f_n be a total, boolean valued function that is total, but not constant on A_n and let $h(x)(n) = f_n(x)$ for $x \in X$.
Then $h(x)$ is total when x is total, and $h(x) = h(y)$ if and only if x and y are consistent.

1.3 A survey of existing results

1.3.1 The theory of transfinite computations

As is quite common with mathematical subjects we will find theorems also in the theory of transfinite computations whose prime aim is a better understanding of the basic concepts. We consider theorems related to the degree-theory and the applications of forcing as such theorems, and it is fair to say that much of the latest developement the theory is introspectivly motivated.

There are however some fundamental results that are of importance for a general analysis of structures of the form $L_\kappa(HC)$.

First we will mention the conceptual clarification that can be gained from the basic definitions.
Ordinals κ may be classified by the concepts of the theory, e.g

$*$ $L_\kappa(HC)$ is admissible but not a limit of admissibles.

$*$ $L_\kappa(HC)$ is admissible and a limit of admissibles.

$*$ $L_\kappa(HC)$ is an E-closure

etc.
One of the basic results is that an E-closure is never admissible. This was essentially proved by Moschovakis [29] when he proved that the semicomputable sets relative to ${}^3E$ is not closed under existensial quantification.

In the case where $L_\kappa(HC)$ is not E-closed we may ask for the complexity of the input leading to a computation of ordinal hight κ, i.e. what is required to make κ E-recursive in some arguments of $L_\kappa(HC)$.

Reflection An interesting group of results are the reflection theorems.

Definition 3 Let $\kappa < \lambda$ be two ordinals. We say that λ reflects on κ if we for every closed Σ_1-statement Φ with HC as the only possible parameter have that

$$L_\lambda(HC) \models \Phi \Rightarrow L_\kappa(HC) \models \Phi.$$

Let κ_0 be the least ordinal not bounded by the length of any computation in ${}^3E$ with natural number inputs, and let C be the local jump of ${}^3E$ defined by $e \in C \Leftrightarrow \{e\}({}^3E)\downarrow$ (i.e. terminates). Let κ_1 be the upper bound of the lengths of computations with input ${}^3E$, C, and natural numbers.

Harrington [18] pointed out that the reflection phenomena are important in computations in higher types, and he proved several basic results about reflection. One simple special case of his results is that κ_1 reflects on κ_0.

A rephrasing of asking for reflection phenomena will be to ask when new Σ_1 facts will be true for $L_{\kappa+1}(HC)$. The lesson from the theory of transfinite computations will be that this will be the case either because κ is a describable closure ordinal, or because something significant in terms of set recursion takes place. The significant something can be that a computation

of length κ exist, or that the verification of nontermination of a computation will exist in $L_{\kappa+1}(HC)$. For further information about global reflection phenomena connected with computations in higher types or with set recursion, see Moldestad [28] or Sacks [39], and for a closer study of local reflection phenomena related to computations in ${}^3E$, see Moldestad [28].

Of course the theory of transfinite computations does not give all the answers in a characterisation and classification of ordinals where Σ_1-statements get true, but it clearly offers both conceptual and operational tools for such a venture.

It is well known to the experts that the path to reflection properties goes via selection theorems. Thus selection theorems is an important part of the theory of transfinite computations.

1.3.2 Representation theorems

A link between the theory of transfinite computations and semantics for types is established via the so called *representation theorems*. The starting point was (independently) the beliefs of Normann and Berger that it should be possible to use transfinite versions of the continuous or countable functionals or to use domains with totality in constructing interpretations of type theory. It turned out that separable domains with totality satisfying density and co-density are very handy in coding complex information, and that a systematic use of this expressive power enables us to prove that the closure ordinal of topological operations like dependent sums and products of parameterised families of domains with totality coincide with ordinals related to transfinite computations.

Definition 4 Let X be a domain with total objects $\bar{X}$, and let R be a subset of $\bar{X}$.
A *positive representation of* R will be a family $\{Y_x\}_{x\in X}$ of domains such that Y_x is a domain with totality $\bar{Y}_x$ for total x, where $\bar{Y}_x$ uniformly satisfies density and co-density, together with a continuous function $\phi \in \prod(x \in X)Y_x$ satisfying

i) If $x \in \bar{X}$ and $x \in R$, then $\phi(x)$ is total in Y_x

ii) If $x \in \bar{X}$ and $x \notin R$, then $h_{Y_x}(\phi(x)) \neq h_{Y_x}(y)$ for all total $y \in Y_x$.

A *negative representation* of R will be a positive representation of the complement of R
A *partial representation* of R will be a relaxation of a positive representation; we only require that Y_x is a domain with totality when $x \in R$, and then that $\phi(x)$ is total

Our definition of representation has its root in Kreisel [22], where a representation theorem for Π^1_k-statements is proved, and where some of our basic methods are used. An important aspect of our representation theorems will be that the total elements of a domain are canonically defined from the description of the domain.

The first representation theorem for transfinite computations occurs in Normann [32], where we also find the first density theorem for a transfinite system of domains with totality. There we prove that we have uniformly positive and negative representations for any subset of $\mathbb{N} \to \mathbb{N}$ computable in ${}^3E$ and a real, and that we have a partial representation for semicomputability in ${}^3E$. In constructing the domains used in the representation we start with the base domain $\mathbb{N}_\perp$ and then inductivly use dependent products of continuously parameterised families of domains with totality.

In Kristiansen and Normann [26] we studied domains with totality obtained by iterating positive inductions. These can in turn be used to construct representations for subsets of $\mathbb{N} \to \mathbb{N}$ definable via iterated positive induction. In Normann [34] this is used to prove a representation theorem essentially for set recursion relative to the next admissible operator, and with HC as basic input.

1.3.3 Representation of structures in general

In section 1.3.2 we defined what we meant by positive and negative representations of a predicate on the total elements of some domain. These concepts can of course be extended to n-ary relations on the total elements. Now, when we say that we have a representation of a relation, we will mean that we have both a positive and a negative representation. In particular it will be clear what we mean by a representation of a relational structure $\mathcal{X} = (\bar{X}, P_1, \ldots, P_n)$, we simply mean that we have representations of all relations involved.

Definition 5 Let $\mathcal{A} = (A, R_1, \ldots, R_n)$ be a relational structure.
A *pre-representation* of $\mathcal{A}$ will be a domain X with totality $\bar{X}$ and relations $I, P_1, \ldots, P_n$ on $\bar{X}$ where

i) I is an equivalence relation (I for 'identity').

ii) For each i, P_i will respect I and have the same arity as R_i,

together with a representation of $I, P_1, \ldots, P_n$ and together with an isomorphism between $\mathcal{A}$ and $(\bar{X}, P_1, \ldots, P_n)/I$.

It is clear that a further analysis of the logical properties of $(\bar{X}, P_1, \ldots, P_n, I)$ will give us information about $\mathcal{A}$. Thus the aplicability of the methods developed in chapter 2 will depend on the existence of structures with pre-representations.

In Normann [34] we use the general representation techniques to show that for certain κ we can construct pre-representations of $L_\kappa(HC)$ using comparative principles of type constructions. We use representations of transfinite computations of length κ to obtain a pre-representation of κ itself, and then a general (and uniform) construction of a pre-representation of $L_\kappa(HC)$ and of all its elements.

1.3.4 Totality without density

Though we will mainly make use of domains with totality, density and co-density in this paper, there are important examples of domains with totality where neither density nor co-density are satisfied. As has been mentioned before, one of the motivations for a systematic study of totality on domains was the prospect of finding semantics for type theory. In that case, a domain interpreting a type will be 'true' if it contains a total object. Thus density would imply that all typed statements are true, and this would not give us an interesting semantics. A first investigation of a hierarchy of domains with totality, where the trivial domain with no total elements is included as a base type, can be found in Normann [33]. The essential technical result there is that every total object of any domain in the hierarchy will respect extential equality, and as a consequence, that extential equality will be an equivalence relation. Waagbø [45] has continued the study of this hierarchy, and he was able to extend it to a model for one version of type theory. In Normann

[35] we will give a further conceptual analysis of domains with totality when density is not taken into account.

1.3.5 Coherence spaces with totality

As an alternative to domain theory we will mention the qualitative domains and coherence-spaces introduced by Girard [16]. Qualitative domains with totality was the theme of Kristiansen [23]. She produced transfinite hierarchies of qualitative domains based on the natural numbers, and closed under sums and products of stable parameterisations of qualitative domains, and she proved density and the analogue of co-density for this hierarchy. One reason for the importance of the results in [23] is that the basic problems like density, representation of computations in ${}^3E$ and modelling positive induction was first solved for constructions of qualitative domains, and the solutions there had a direct impact on the methods used for the analogue results for domains. Another reason is that Girard's coherence space semantics for System F [16] was directly influenced by his work on Π^1_2-logic, focusing on stability and commutation with pullbacks and direct limits. It is possible that a more genuine generalisation of β-logic will require the results and methodology from [23]. The results from [23] have been or will be published in Kristiansen and Normann [24, 25, 26].

2 Transfinite proof systems

2.1 ω-logic and Σ^1_1-logic

ω-logic is the classical example of an infinite proof system used to investigate a finitary proof system, e.g. Peano Arithmetic (PA).
One advantage of ω-logic is that it satisfies cut-elimination, and cut free proofs often contain more information about the proved statements than proofs with cuts. This was first used by Kreisel [21] to analyse provability in PA, for a recent and elegant exposition, see Buchholz and Wainer [6].

Girard [14] constructed β-logic or Π^1_2-logic with a similar use in mind. A β-proof is a uniform collection of α-proofs where α is an ordinal, and a statement Φ in a special sort $(\Omega, \leq)$ has a β-proof if and only if Φ is true in all models where $(\Omega, \leq)$ is interpreted as a well ordering. As Girard points

out, this corresponds to truth in all β-models, i.e. all models of 2'nd order aritmetic where well foundednes is absolute.

β-logic is called Π^1_2-logic because the basic concepts involved, those of 'β-proof' and 'truth in all β-models' are complete Π^1_2. By analogy, ω-logic may be called Π^1_1-logic. Girard [15] actually use the terminology Σ_0 - logic for ordinary finitary logic and Π^1_1 - logic for ω-logic.
In cooperation with T. Engen the author constructed an intermediate logic, Σ^1_1-logic. The proof objects are computable, possibly non-wellfounded, proof trees using the ω-rule such that there are no infinite hyperarithmetical branches. The completeness theorem for this logic states that a statement has a proof if and only if it is true in all hyperarithmetical ω-models. Using the Spector-Gandy theorem, (see Spector [43], Gandy [12] or Rogers [38]), we see that the concepts of truth and proof both are complete Σ^1_1.

Σ^1_1-logic will satisfy cut-elimination. The idea is that cut-elimination can be viewed as a top-down procedure (with the proved statement at the top) where we linearise branchings due to the use of the cut rule. Since this branching is binary, we do not introduce hyperaritmetical branches in this process. The details are written out in Engen [7] (in Norwegian).

The strict uniformity in the concept of a β-proof will be relaxed to continuous dependence of the parameter. This will be carried out in the next section. There we will give the technical definitions and prove a general cut-elimination theorem for this kind of proof systems. In the sections to follow we will see how the implementation of the induction axiom of PA in ω-logic can be generalised and how our definition can be used to describe absolutenes relative to rather complex subsets of $\mathbb{N} \to \mathbb{N}$.

2.2 X-proofs

2.2.1 Proof-trees with continuous branching

Let X be a separable domain and let $\bar{X}$ be a subset of X. We will call the elements of $\bar{X}$ *total*, and we will assume that $\bar{X}$ satisfies density and co-density as defined in Definition 1.

We will let $\bar{X}$ be the ground set of a first order structure $\mathcal{X}$, with functions $\bar{f} : \bar{X}^n \to \bar{X}$ and predicates $\bar{R} \subseteq \bar{X}^n$.
Each $\bar{f}$ will be the restriction of a continuous $f : X^n \to X$, and for each $\bar{R}$ we will have disjoint open subsets R^+, and R^- of X^n such that R^+ and R^-

are complementary on X^n with $\bar{R} = R^+ \cap \bar{X}^n$.

We will assume that $\mathbb{N}$ is an identifiable subset of $\bar{X}$ via a unary predicate in $\mathcal{X}$ and that the map $h : \bar{X} \times \mathbb{N} \to \mathbb{N}$ is one of the functions in $\mathcal{X}$, and we assume that equality on $\mathbb{N}$ is one of the binary relations in $\mathcal{X}$.

Now $\mathcal{X}$ will be a structure for a first order language L with variables $x_1, x_2, \ldots$. For the sake of notational complexity we will not distinguish between the symbols of the language and their interpretations.

The language L will be extended with *predicate variables* Q of fixed arity. Any subset of $\bar{X}^n$ will be accepted as an interpretation of an n-ary predicate variable. For the sake of simplicity we will assume that we only have one predicate variable Q, and that it is unary.

Definition 6 a) A *structure* $(\mathcal{Y}, \mathcal{Q})$ will be a pair where $\mathcal{Y}$ is a substructure of $\mathcal{X}$ and $\mathcal{Q}$ is any subset of the ground set of $\mathcal{Y}$.

b) Let Φ be a closed formula in the language L, Q.
Φ is *valid* if $(\mathcal{Y}, \mathcal{Q}) \models \Phi$ for all structures $(\mathcal{Y}, \mathcal{Q})$ with $\mathcal{Q}$ as the interpretation of Q

We will develope a proof-system for this notion of validity.

Remark If we view $\bar{X}$ as a topological space with the topology inherited from X, any subset of $\bar{X}^n$ that is both closed and open can be used as an interpretation of an n-ary predicate symbol. This is a consequence of a basic lifting theorem that will appear in Normann [35]. In most of the relevant applications there are more general lifting-theorems ensuring that any continuous function $\bar{f} : \bar{X}^n \to \bar{X}$ can be used as an interpretation of an n-ary function symbol.

Example

$$X = ((\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp) \oplus (\mathbb{N}_\perp \to \mathbb{N}_\perp) \oplus \mathbb{N}_\perp$$

The total objects will be the total objects of type 2, 1 and 0 resp. There will be three unary predicates corresponding to the three types. and there will be two binary predicates representing equality and ordering on $\mathbb{N}$. On this set it is natural to consider the two evaluation-functions as functions of the structure. Moreover we may include any set of computable functions on $\mathbb{N}$ in the structure.

We are now ready to define X-logic. We will use a traditional variant of sequent calculus.
We will use positive and negative literals as our base formulas. The set of formulas will be closed under the connectives $\wedge$ and $\vee$ and under the quantifiers $\forall x_i$ and $\exists x_i$ in the usual way. We let $\neg$ and $\rightarrow$ be defined connectives in the usual way.
Our sequents will be sets $\Phi_1, \ldots, \Phi_n$ of formulas, where we use the disjunction of the formulas as the interpretation of the sequent. Our logic will deal with sequents with instansiations from $\bar{X}$.

Definition 7 The underlying language with formulas, sequents etc. is considered as a flat domain.

* An *instansiation* will be a set $\{\alpha_k\}_{k\in K}$ from X where K is a finite subset of $\mathbb{N}$. We will denote such instansiations by $\vec{\alpha}_K$.

* We will consider the set of all instansiations as the dependent sum of a parameterised family of domains, where the domain of parameters is the flat domain with the finite subsets of $\mathbb{N}$ as maximal objects.

* A sequent Γ is *relevant* for a finite set K if all free variables in formulas in Γ have their indices in K.
  The set of sequents relevant for a given instansiation can be considered as a subdomain of the flat domain of all sequents, and, trivially, the corresponding parameterisation over the domain of finite sets in $\mathbb{N}$ will be continuous in the sense of domain theory.

Below we will give our definition of an $\vec{\alpha}_K$-proof. The set of $\vec{\alpha}_K$-proofs will be a domain, and if we view them as a family of domains parameterised over the domain of instansiations, the parameterisation is continuous in the sense of domain theory. What we actually do below is to write a system of domain equations for this parameterisation. As a consequence we will have to view the bottom element $\perp$ as a proof object. At the end we will only be interested in proof objects for total instansiations $\vec{\alpha}_K$ that are well founded when branchings are resticted to total objects.

Definition 8 We will always assume that the sequent in question is relevant for the index set of the instansiation.
If

$$\Gamma = \Phi_1, \ldots, R(\vec{t}(\vec{x})), \ldots, \Phi_n$$

and $\vec{t}(\vec{\alpha}_K) \in R^+$, then $((\Gamma), A)$ is an $\vec{\alpha}_K$ proof corresponding to an *axiom*.
Here A (short for 'Axiom') is just some atom in a flat domain.
Moreover, if $\Gamma = \Phi_1, \ldots, \neg R(\vec{t}(\vec{x})), \ldots, \Phi_n$, and $\vec{t}(\vec{\alpha}_K) \in R^-$, then $((\Gamma), A)$ is an $\vec{\alpha}_K$-proof.
The axioms of the theory are essentially the diagram of the structure $\mathcal{X}$, and thus in reality nonlogical axioms. In order to handle the predicate Q we need some form of the logical axiom $\Gamma, Q(t), \neg Q(t)$. For technical reasons we will use a rule of deduction instead of this axiom. The technical reason is that we must respect equality between terms, even though equality is not a part of our language, and even though equality is not decidable.

a) We let the expression $s = t$ be short for the formula

$$\forall n(h(s)(n) = h(t)(n))$$

b) If $((\Gamma, s = t), P)$ is an α_K-proof, then

$$((\Gamma, Q(s), \neg Q(t)), E, (\Gamma, s = t), P)$$

is an α_K-proof, where E signifies that we have used the *rule of equality*.

The $\wedge$-rule:
If $((\Gamma, \Psi_1), P_1)$ and $((\Gamma, \Psi_2), P_2)$ are two $\vec{\alpha}_k$-proofs, then

$$((\Gamma, \Psi_1 \wedge \Psi_2), (\wedge, \Psi_1, \Psi_2), ((\Gamma, \Psi_1), P_1), ((\Gamma, \Psi_2), P_2))$$

will be an $\vec{\alpha}_K$-proof. The $\vee$-rule and the cut-rule are treated in a similar fashion.

$\exists$-rule:
If $((\Gamma, \Psi(x_i/t)), P)$ is an $\vec{\alpha}_K$-proof, then

$$((\Gamma, \exists x_i \Psi), (\exists, t), ((\Gamma, \Psi(x_i/t)), P))$$

is an $\vec{\alpha}_K$-proof.

In order to describe the $\forall$-rule we need a bit more notation.
By $\vec{\alpha}_K(\alpha_i/\beta)$ we mean the instansiation obtained by replacing α_i with β as an element with index i if $i \in K$, and the instansiation obtained by adding β as an element with index i if $i \notin K$. We then have to add i to K.
If F is continuous such that $F(\beta) = ((\Gamma, \Psi), P_\beta)$ is an $\vec{\alpha}_K(\alpha_i/\beta)$-proof for all $\beta \in X$, then $((\Gamma, \forall x_i \Psi), \forall, F)$ is an $\vec{\alpha}_K$-proof.

We will now isolate the genuine proof objects:

Definition 9 If $\vec{\alpha}_K$ is an instansiation from $\bar{X}$ we define the *well-founded* $\vec{\alpha}_K$-proofs as follows:

i) Axioms are well-founded proofs.

ii) In the cases of the $\wedge$-rule, the $\vee$-rule, the cut-rule, the $\exists$-rule and the rule of equality, the immediate subproofs have to be well-founded.

iii) In the case of the $\forall$-rule, $F(\beta)$ has to be a well-founded $\vec{\alpha}_K(\alpha_i/\beta)$-proof for all $\beta \in \bar{X}$.

This definition gives us a family of well-founded proof trees where we have used a continuous $\bar{X}$-rule, and we will use the term $\bar{X}$*-proof* for these proof trees.
The well-founded proof trees will be the total objects in some parameterised family of domains with totality defined via a strictly positive induction with cross references to other domains in the parameterisation. Such sets are defined as the total elements of *typestreams*, see Kristiansen and Normann [26] or Normann [35]. A type-stream is essentially a domain defined via a top-down description of how the domain is composed, where the total objects are defined via a bottom-up induction.

2.2.2 Soundnes and completenes

The soundnes-theorem is trivial, and the proof is left for the reader:

Theorem 1 *Let $\vec{\alpha}_K$ be a total instansiation and let the sequent Γ be relevant for $\vec{\alpha}_K$.*
If there is a well-founded $\vec{\alpha}_K$-proof for Γ, then $\Gamma(\vec{\alpha}_K)$ will be true in all structures $(\mathcal{Y}, \mathcal{Q})$ with $\vec{\alpha}_K$ as elements.

We will now prove the converse, the completenes theorem:

Theorem 2 *Let Φ be a closed formula.*
If Φ is true in every structure $(\mathcal{Y}, \mathcal{Q})$, then Φ will have an $\bar{X}$-proof.

Proof
We will adjust one of the standard metods for proving the completenes theorem.
First we define the *reduction tree* for Φ by analysing Φ and its subformulas. Without loss of generality we will assume that whenever we want to reduce a subformula $\exists x\Psi$ there will at least be one term available. We will also assume that Φ is a closed formula, since our logic deals with closed instances. There will be no problem extending the argument to some fixed closed instance of a formula with free variables.
The definition of the reduction tree is top-down, where we ad new nodes as the tree grows downwards. Each new node will be a reduction of some node above, and by a standard book-keeping device we can ensure that all required reductions can be carried out in all relevant branches.
Φ will be the top node of the tree.
Reduction of $\forall x_i\Psi$:
If a branch contains $\forall x_i\Psi$, then at a node further down in the tree it will contain Ψ.
If a branch contains a node $\exists x_i\Psi$ and if t is a term in variables occuring free somewhere in the branch, then the branch also contains $\Psi(t)$.
If a branch contains $\Psi_1 \vee \Psi_2$ then it will also contain both Ψ_1 and Ψ_2.
If a branch contains $\Psi_1 \wedge \Psi_2$, then somewhere below there will be a branching with Ψ_1 as the next node in one branch and Ψ_2 in the other.
In the standard construction we will stop a branch when two literals $Q(t)$ and $\neg Q(t)$ occur in the branch. Since we are dealing with substructures of $\mathcal{X}$, we may get two literals $Q(t)$ and $\neg Q(s)$ where s and t are syntactically different, but where the interpretations under an instansiation are equal. This is undecidable, and we will rely on the rule of equality to get around this obstacle.
In the construction of the reduction tree we will require:
If a branch contains two literals $Q(t)$ and $\neg Q(s)$ then somewhere below there is an $\mathbb{N}_\perp$-branching where the next formula in the n'th branch is
$h(t)(n) = h(s)(n)$.

For any node in the tree we will now consider the sequent consisting of this node and all the formulas above the node in the tree. Considering all possible total instances of such sequents, we get a new tree by essentially replacing any reduction from $\forall x_i\Psi$ to Ψ to a branching into all $\Psi(\beta)$ for β ranging over

$\bar{X}$.
Finally we prune this new tree by cutting off the tree wherever we get an instance of a sequent that is an axiom.

There are two cases:

Case 1 There is an infinite branch in this pruned tree.

Case 2 There is no infinite branch in this pruned tree.

In case 1 we see that we may construct the substructure of $\mathcal{X}$ generated from the total elements used in the instansiations of the branch. If $Q(s)$ and $\neg Q(t)$ occurs in the branch, there will be an atomic statement $h(t)(n) = h(s)(n)$ in the branch which will be false under the instansiation. Thus we may define $\mathcal{Q}$ so that all literals in the branch containing Q will be false. By a standard induction on the complexity of a formula we then see that any formula occuring in the branch will be false under the instance in question.

In case 2 we see by a standard contrapositive argument that Φ will be valid. We will use recursion to construct an $\vec{\alpha}_K$-proof for Γ whenever $\Gamma(\vec{\alpha}_K)$ is one of the instances of a sequent of the branch.
First we will construct a continuous function into the flat domain
$\{$ Yes , No $\}_\perp$ deciding (when decidable) wether a node in the tree is an end node in the pruned tree. We will describe this function informally, but technically we will use the fix-point of a recursive operator on the underlying domain.
If the sequent contains a literal $R(\vec{t})$ or $\neg R(\vec{t})$, we must know the truth value of this literal. If the literal is true and we know that no subsequent is an axiom, we give output 'yes'. If the literal is false and we know that no subsequent is an axiom we give output 'no'. If the truth value of the literal is undefined (i.e. neither R^+ nor R^- contains the interpretation of the terms) we just have to say that we do not know, and then of course, for any node further down in the tree we also have to say that we do not know.

Now, to each node in the nonpruned tree we will associate a proof object for the sequent associated with that node.
If it is undecidable wether the sequent is an axiom or not, we just use the empty proof-object, i.e. the $\perp$-element of the domain of proofs.
If the sequent is an axiom, we restrict ourselves to the first node in the branch where this is the case, and use this axiom as the proof object.

If the sequent is not an axiom we construct a proof object for the sequent from the proof objects of the immediate successor(s) depending on which reduction we have carried out at this node. The only non-traditional case is when we analyse the two literals $Q(t)$ and $\neg Q(s)$. Uniformly in n we have a proof-object for the sequent $\Gamma, Q(t), \neg Q(s), h(s)(n) = h(t)(n)$, so by the ω-rule (which is a special case of the X-rule), we get a proof object for $\Gamma, Q(t), \neg Q(s), s = t$. Then by the rule of equality we get a proof object for $\Gamma, Q(t), \neg Q(s)$.
Since this is uniform in the reduction tree there is no problem in knowing continuously what to do. Thus we have given an accurate description of the proof-object.

Finally we observe that for total instansiations we will always be able to decide wether a sequent is an axiom or not, and it then follows by induction on the rank in the pruned tree that for every sequent in that tree we construct a well-founded proof object. Thus we end up by constructing a well-founded proof for Φ.

This ends the proof of the completenes theorem.

2.2.3 Cut elimination

Though the proof objects constructed in the proof of the completenes theorem both are cut free and computable (modulo a representation for $\mathcal{X}$), there is little information to be found in these proofs. It is to be expected that if we translate some simpler proof-system into the system of $\bar{X}$-proofs, we will use cut in the translation. As an example we will see how to transform proofs by induction to $\bar{X}$-proofs when $\bar{X}$ is inductivly defined. We will prove a cut elimination theorem for $\bar{X}$-logic. The growth of complexity will be as with other cut elimination theorems. We do not state this as a part of our theorem.

Theorem 3 *There is a continuous function c that to an $\bar{X}$-proof for an instance of a sequent gives a cut free proof for the same instance of the same sequent.*

We will use the conventional proof of cut elimination leading to estimates of complexity. We will have to pay special attention to the rule of equality. The standard proof actually gives us some equations for the cut free proof, so we prove cut elimination by solving these equations over the domain of proofs.

In the standard proof of cut elimination we assume that we have eliminated all cuts above one occurence of the rule, and then show how one use of cut can be pushed up in the proof tree. Thus we will define an assisting function e that will eliminate cut when there is no occurences of cut above. We also define e by solving the equations for it.

The equation for c will consist of three main cases:

In case of an axiom, c will be the identity.

In case of any rule exept the cut-rule, we let c commute with the rule.

In case of the cut-rule we first apply c on the two subproofs and then apply e.

In the proof objects we have coded in the hard way what rules are used and what the subproofs are, so there is no problem in distinguishing between the various cases. This will also be the case for the recursion equation for e described bellow.

The equation for e is the heart of the cut-elimination proof. We must consider the rules used just above the cut-rule. We have the following 3 cases:

Case 1 The formula just introduced on the lefthand side is not the cut formula:
Let e commute in its first variable with the last rule used on the lefthand side. In the case that we have an axiom not involving the cut-formula, e will observe that the main sequent is an axiom, and stop the elimination.

Case 2 The formula just introduced on the lefthand side is the cut formula, but this is not the case on the righthand side:
Then e will commute in its second variable with the rule in analogy with the case above.

Case 3 The last formulas introduced on both sides are the cut formulas:
What we do next will then depend on the cut-formula.
In the case of $\wedge$, $\vee$ there are no problems.
In the case $C = \exists x D$ and $\neg C = \forall x \neg D$, the standard method is to use a sub-proof for $D(x/t)$ and the proof we get for $\neg D(x/t)$ when we replace x

by t in the proof tree for $\neg D(x)$. There are no problems in describing a continuous function that given an $\vec{\alpha}_K(\alpha_i/\beta)$-proof for $\neg D$ gives us uniformly in β an $\vec{\alpha}_K$-proof for $\neg D(x_i/t)$. Using this continuous transcription of proofs it is clear what e will do in this case.
Our final case is when the cut formulas are literals.
It is impossible that the literals are of the form $R(t)$ or $\neg R(t)$ since these must be introduced as axioms, and not both of them will be axioms at the same time. So we are left with the case

$$\dfrac{\dfrac{\dfrac{P_1}{\Gamma, s=t}}{\Gamma, Q(s), \neg Q(t)} \quad \dfrac{P_2}{\Delta, \neg Q(s)}}{\Gamma, \Delta, \neg Q(t)}$$

By a recursive operator on the proof P_1 for $\Gamma, s = t$ we can reconstruct a cut-free proof for $\Gamma, h(s)(n) = h(t)(n)$ uniformly in n. Now, by recursion on the proof P_2 of $\Delta, \neg Q(s)$ we attemt to construct a proof of $\Gamma, \Delta, \neg Q(t)$ by systematically substituting t for s, adding Γ to each line and otherwise use the same rules. Occationally in a subproof we will introduce $\neg Q(s)$ via the rule of equality, and then somewhere further up in the tree we may find an axiom $\Delta', h(u)(n) = h(s)(n)$ which we attemt to rewrite to $\Gamma, \Delta', h(u)(n) = h(t)(n)$. If the rewritten sequent is not an axiom, we must have $h(s)(n) \neq h(t)(n)$. But then $\Delta', h(u)(n) = h(t)(n), h(s)(n) \neq h(t)(n)$ is an axiom, and we can use cut with the provable $\Gamma, h(s)(n) = h(t)(n)$ to obtain locally a proof of $\Gamma, \Delta', h(u)(n) = h(t)(n)$. Our cut-elimination process e will involve this rewriting together with the elimination of the new cuts constructed.
This ends our construction. It is now trivial by recursion on the well-founded total part of the original proof tree to show that these operators do what they are designed to do, they produce well-founded cut-free proof trees.

Remark
Based on the soundnes and completenes theorems there is a much simpler proof of the cut-elimination theorem as stated. We simply observe that the cut-free proof constructed in the proof of the completenes theorem depends continuously on the formula analysed, and that this can be extended to any valid interpretation of any sequent. Thus it is the actual estimates of complexity that is of importance with the above proof. The new thing here is the observation that the standard proof of cut elimination leads to domain equations which then can be solved, so that our proof system will be closed under the standard cut elimination process.

2.3 The logic of parameterised families of domains

We introduced $\bar{X}$-logic above, and proved soundnes, completenes and cut elimination. The constructions used are so uniform that if we have a nice parameterisation $\{Y_x\}_{x\in X}$ with a subparameterisation $\{\bar{Y}_x\}_{x\in\bar{X}}$ of domains with totality, then we have a uniform notion of $\bar{Y}_x$-proof with the corresponding soundnes, completenes and cut elimination. For this to work exactly as stated, we will have to require that each Y_x are extended to structures for one common signature in a uniform way. There will, however, be cases where some structure on X is relevant, and also where there are operators mapping elements of some Y_x into some other $Y_{x'}$. The actual proofs of the three basic theorems will be more or less the same as in the simple case, but formulating the languages and defining interpretations and proofs in the desired generality is notationally complex. In this section we will define languages suitable for investigating parameterised families of domain-based structures with cross-reference.

In this section, a *basic structure* will consist of a domain X, a parameterisation $\{Y_x\}_{x\in X}$ and the dependent sum $\sum(x \in X)Y_x$, where we let the pairing (x, y) and the projections π_0 and π_1 be a part of the basic structure. A basic structure will be equiped with a totality, i.e. a set $\bar{X} \subseteq X$ and for each $x \in \bar{X}$, a subset $\bar{Y}_x \subseteq Y_x$. We will assume uniform density and codensity. For the details of this paper, we will not need the precise technical definition given by Berger [5].
A *structure* will be a basic structure with totality together with

i) Total, continuous functions and relations on $\bar{X}$ as in the previous section.

ii) Some total, continuous functions

$$f : (\sum(x \in X)Y_x)^n \to \sum(x \in X)Y_x$$

iii) Some uniformly total families of relations $(R_x^+, R_x^-) \subseteq Y_x^n$.

We will now describe the language.
First we introduce an infinite list of variables x_i of sort X.
This, and the rest of our definition will give us some terms of sort X.
To each term t of sort X we introduce a list of variables x_i^t of sort Y_t.
Using pairing and the functions on the dependent sum we get terms of sort

$\sum$ (short for the dependent sum), and using projections we get new terms of sort X and $Y_{\pi_0(t)}$ resp.
Using the obvous cancelation rules for pairing and projections we may identify some terms of sort X, and then some terms of sorts Y_{t_1} and Y_{t_2} for equivalent terms t_1 and t_2. This equivalence is determined purely at language level and has nothing to do with the actual functions and relations used. We will thus identify equivalent terms in our language. This means in particular that if t_1 and t_2 are equivalent terms, then the variables $x_i^{t_1}$ and $x_i^{t_2}$ are the same.
We will accept predicate variables Q of type X or variables $\{Q_x\}_{x\in X}$ of type $\{Y_x\}_{x\in X}$.

Given this system of terms and predicates with sorts we may define the formulas of the language. Literals are defined in the usual way, where the sorts of the predicates and the terms must match. The sort depends only on the syntax, so this is meaningful. We close the set of formulas using $\wedge$, $\vee$, $\forall$ and $\exists$ in the usual way.
In the case of $\forall x_i^t\Psi$ and $\exists x_i^t\Psi$ we will assume that no variable free in t will be bounded in Ψ.

This language is of course purely syntactical, and as before the ingredients can be viewed as the maximal elements of some flat domain. In defining $\bar{X}$-logic we let an instansiation just be a finite set of domain objects, and we isolated the sets of formulas relevant for such sets. Here we have to be a bit more careful.

Definition 10 Let K be a finite set of variables. We say that K is *closed* if whenever a variable $x_i^t \in K$, then all variables occuring in t will also be in K.

Clearly, every finite set of variables will be contained in a closed, finite set of variables.

Definition 11 i) A legal interpretation of a variable x_i of sort X will be an element of X.

ii) A legal interpretation of a variable x_i^t will consist of legal interpretations $\vec{\alpha}$ of the variables occuring in t together with an element of $Y_{t(\vec{\alpha})}$.
Normally we will only refer to the last object as the interpretation of the variable.

iii) If K is a closed set of variables, an *instansiation* over K will be a set of legal interpretations of the variables in K such that two interpretations of the same variable will coincide.

We may use the same definition as before of a formula being relevant for an instansiation. The set of instansiations will form a domain in the obvious way, using dependent sums in the case where the range of one variable will depend on the interpretation of others.
We may further generalise the domain of proof-objects in the analogue way including both the X-rule and uniformly the Y_t-rules. Again the wellfounded proofs can be seen as the total objects in some type-stream, so they themselves form a domain with totality. The proofs of soundnes, completenes and cut-elimination will now esentially be the same as above. We do not go into details here.
In many applications of this generalised case, the total elements $\bar{X}$ will be of higher complexity than all the $\bar{Y}_x$, and it may be that it is the limit of the complexities of the $\bar{Y}_x$'s that is of interest. For the rest of this section we will assume that we work with the language and logic of a parameterisation.

Definition 12 A formula Φ is *bounded* if all quantifiers are of the form $\forall x_i^t$ or $\exists x_i^t$.

We have defined the concept of a proof for closed instances of formulas. It is to be expected that different instances of the same formula will have different proofs. In this context it is however natural to consider classes of proofs where the proof of one instance of a formula depends continuously on the instansiation. We formulate this in the following definition:

Definition 13 Let Φ be a bounded formula with free variables of sort X.
A *uniform, locally bounded proof* for Φ will be a continuous function that to each instance $\vec{\alpha}$ of the free variables $\vec{x}$ gives a proof of $\Phi(\vec{\alpha})$ that does not use the X-rule.

We then have that a bounded formula is valid, i.e. all instances are true in all substructures containing the instansiation, if and only if the formula has a uniform, locally bounded proof. The proof of this will again only use the methods of our completenes theorem. Moreover, it is easy to see that the cut elimination procedure applied to a uniform, locally bounded proof will give us a uniform, locally bounded proof.

2.4 X-logic and strictly positive induction

In this section we will see how formal proofs using an induction rule can be translated to proofs of the nature studied above. Essentially we show that the translation of Peano Arithmetic to ω-logic can be generalised to domains constructed via strictly positive inductions with cross references over a parameterisation. As a generic example we will discuss the natural interpretations of Per Martin-Löf's W-types (Martin-Löf [27]) in some details. Our methods easily extend to general strictly positive induction.

We will let $A, \bar{A}, \{B_a\}_{a\in A}, \{\bar{B}_a\}_{a\in\bar{A}}$ be a parameterisation of domains with totality.
We let W be the solution of the equation

$$W = \sum_{a\in A}(B_a \to W)$$

The elements of W can be considered as trees with branchings over B_a. $\bar{W}$ will be the well founded trees where we only consider branchings over $\bar{B}_a$ for $a \in \bar{A}$. This is of course the correct interpretation of Martin-Löf's W-type [27], see also section 1.2.2.

The domain W with totality $\bar{W}$ is an example of a type-stream as defined in [26] or [35].
The inductive definition of $\bar{W}$ will provide us with a valid rule of induction. Before formulating it in our context we will describe the natural language for W. What we need are two functions I and E. I gives the index and E the evaluation of the tree in the following way.

$I : W \to A$ with $I(a, f) = a$

$E : \sum(w \in W)B_{I(w)} \to W$ with $E((a, f), b) = f(b)$.

If $A, \{B_a\}_{a\in A}$ has some internal structure we include this, I and E in the manysorted structure $(A, \{B_a\}_{a\in A}, W)$. It is in the language of this structure that we will formulate the induction rule.

Definition 14 a) The induction rule for $\bar{W}$ is the following

$$\frac{\forall w \in \bar{W}\forall b \in \bar{B}_{I(w)}(\Phi(E(w, I(w))) \to \Phi(w))}{\forall w \in \bar{W}\Phi(w)}$$

b) We include the possible use of this rule in the definition of an $\vec{\alpha}_K$-proof in the canonical way.

c) A formula in the language of W , A , $\{B_a\}_{a\in A}$ is *provable by induction* if there is a total continuous proof tree using the A-rule, the B_a-rules and this new rule of induction.

We will now see that this induction rule can be reduced to the W-rule in the usual way:

Lemma 1 *There is a continuous function that to an $A, \{B_a\}_{a\in A}$-proof with induction for a formula Φ gives us an $A, \{B_a\}_{a\in A}, W$-proof for Φ not using induction.*

Proof
This argument is standard. By induction on the proof tree, we transcribe the proof translating the use of the rule of induction to a multiple use of the W-rule and the cut-rule.
In the induction step we establish individual proofs for $\Phi(w)$ for each $w \in \bar{W}$ using the assumption that we have individual proofs for all the predecessors, the cut rule and the induction-free proof of the assumption of the induction rule.
Since every part of the transcription is locally continuous, we will use the fixpoint theorem for domains to define the transcription as a continuous operator, and then use transfinite induction to prove that it maps well founded proofs to well founded proofs.
This ends our proof of the lemma.

It is of course possible to estimate the ordinal hight of the transcript of a proof. If λ is the length of the proof with induction and κ is the rank of $\bar{W}$, then an estimate for the induction free transcript of the proof will be $\kappa \cdot \lambda$.

2.5 Representable structures

In section 1.3.3 we introduced the concept of a pre-representation. In Normann [34] we show how to construct pre-representations of a variety of structures.

In this section we will see how we may use the existence of a representation together with the continuous proofs to construct logics for structures with

representations. In the first subsection we will construct this logic, and in the second subsection we will discuss some examples and applications.

2.5.1 The logic of a representable structure

In this section we will consider a domain X with totality $\bar{X}$ and some additional structure that is not topologically nice in itself but that can be represented in other domains with totality in our standard way. We will restrict ourselves to the case of one unary relation R on $\bar{X}$, but our method can without difficulty be applied to any situation where we have a finite set of relations of finite arity. Our methods do not cover structures with functions. In addition we will ad a unary predicate variable Q that can be interpreted as arbitrary subsets of $\bar{X}$.

We will let R have a representation

$$(\{A_x\}_{x\in X}, \phi, \{B_x\}_{x\in X}), \psi)$$

where $\{A_x\}_{x\in X}$ and $\{B_x\}_{x\in X}$ are parameterised families of domains with uniform density and co-density, and where ϕ and ψ are positive representations of R and $\neg R$ resp. in the sense of section 1.3.2 .

We will see how a statement about R can be translated to a formula in a suitable continuous structure over a family of domains with totality.
Our manysorted structure will consist of X, of the parameterisations of A_x and B_x over X, and of the domain $\mathbb{N}_\perp$ with equality. In addition we will include the following functions in our structure:

i) $f_1 : \sum(x \in X)(A_x \times \mathbb{N}_\perp) \to \mathbb{N}_\perp$ defined by
$f_1(x, a, n) = h_{A_x}(a)(n)$.

ii) $f_2 : \sum(x \in X)(B_x \times \mathbb{N}_\perp) \to \mathbb{N}_\perp$ defined by
$f_2(x, b, n) = h_{B_x}(b)(n)$.

iii) $f_3 : X \times \mathbb{N}_\perp \to \mathbb{N}_\perp$ defined by
$f_3(x, n) = h_{A_x}(\phi(x))(n)$

iv) $f_4 : X \times \mathbb{N}_\perp \to \mathbb{N}_\perp$ defined by
$f_4(x, n) = h_{B_x}(\psi(x))(n)$.

In the definition below we will use the h-functions without indices. The correct index is clear from the context, and any use of the symbol h can be replaced by a use of one of the f_1 to f_4.

Definition 15 Let Φ be a 1'st order statement in the relation R and the predicate Q where all quantifiers range over $\bar{X}$.
Let $\hat{\Phi}$ be the statement with quantifiers over $\bar{X}$ and over $\bar{A}_x$ and $\bar{B}_x$ where x is total in X obtained by replacing

$x \in R$ by $\forall b \in B_x \exists n(h(b)(n) \neq h(\psi(x))(n))$

$x \notin R$ by $\forall a \in A_x \exists n(h(a)(n) \neq h(\phi(x))(n))$

We say that Φ is provable (relative to the representation) if *the transcript* $\hat{\Phi}$ has a continuous, well founded proof in the sense of this paper.

Lemma 2 *Let $\mathcal{X} = (\bar{X}, R)$ be a structure with a representation as above. Let Φ be a closed formula in the language with R and a predicate variable Q and let $\hat{\Phi}$ the transcript.*
Then Φ is valid over all substructures of $\mathcal{X}$ if and only if $\hat{\Phi}$ is valid over all substructures of the extended structure.

Proof
If $\hat{\Phi}$ is valid, then it is easy to see that Φ will be valid, any substructure of $\mathcal{X}$ can be exteded in a maximal way to a substructure of the extended structure, and then the translation from Φ to $\hat{\Phi}$ will be faithful.
To prove the other way, we observe that if we translate Φ to $\hat{\Phi}$, and then interpret $\hat{\Phi}$ over a substructure, what we do is to interpret the literals in Φ as larger sets than the true interpretations. Since Φ is positive in the literals, if Φ is true in the true interpretation, it will remain true under this extended interpretation of the literals. But the translation to $\hat{\Phi}$ will be faithful with respect to this extended version, so $\hat{\Phi}$ will be valid.

We then have the following completenes theorem:

Theorem 4 *Let $\mathcal{X} = (\bar{X}, R)$ be a structure with a representation as above. Let Φ be a closed formula in the language of $\mathcal{X}$.*
Then Φ is provable if and only Φ is valid in all substructures $\mathcal{Y}$ of $\mathcal{X}$.

2.5.2 Examples and discussion

In this section we will discuss some examples of the logics of the previous section.

Π^1_2-logic First let us see how the β-logic of Girard [14] can be rephrased in this setting. We do not claim that what we do gives an improvement or even an equally good treatment of β-truth as the original β-logic, we only aim at illustrating our concepts via a known example.

Let Π be a complete Π^1_1-subset of $(\mathbb{N} \to \mathbb{N})$, e.g.

$$g \in \Pi \Leftrightarrow \forall f \exists n(f(n) = f(n+1) \vee g(f(n), f(n+1)) \neq 0)$$

where we asssume some standard pairing of $\mathbb{N}^2$ into $\mathbb{N}$.
We let Π denote the actual set,and we let $\mathcal{N}$ be the structure $(\mathbb{N} \to \mathbb{N}, \Pi)$. We let π be the formula defining Π.
For each set $A \subseteq \mathbb{N} \to \mathbb{N}$, we let $\mathcal{A}$ be the substructure of $\mathcal{N}$ with A as its groundset.

Definition 16 $A \subseteq \mathbb{N} \to \mathbb{N}$ is called a *β-set* if $\mathcal{A} \models \pi = \Pi$

As a trivial fact we get that A is a β-set if and only if

$$\mathcal{A} \models \forall g(\pi(g) \to g \in \Pi).$$

We then get

Lemma 3 *Let Φ be a 1'st order statement about $\mathbb{N} \to \mathbb{N}$ with an extra predicate variable Q. Then the following are equivalent.*

i) *Φ is true for all β-sets*

ii) *The statement*

$$\forall g(\pi(g) \to g \in \Pi) \to \Phi$$

is true in all substructures of $\mathcal{N}$.

iii) *The transcript of $\forall g(g \in \pi \to g \in \Pi) \to \Phi$ relative to a representation of Π has a computable, continuous proof.*

Since the predicate Π does not occur in Φ we can give a complete description of how this predicate is used. It is easy to see that all literals involving Π will be of the form $t \notin \Pi$. In order to carry out this example further, we will construct a simple representation of Π.

Let $X = (\mathbb{N}_\perp \to \mathbb{N}_\perp) \to \mathbb{N}_\perp$ with $\bar{X}$ as the canonical set of total objects. Let $\phi(g) = \lambda f \mu n (f(n) = f(n+1) \vee g(f(n), f(n+1)) \neq 0)$. Then $\phi(g)$ is total if and only if $g \in \Pi$. Moreover $h(\phi(g))$ will be total for all total g since in order to compute $h(\phi(g))(n)$ for some n we only have to find $\phi(g)(f)$ for some f that is almost constant.

Thus we may rephrase the literal $t \notin \Pi$ by

$$\forall x \in \bar{X} \exists n (h(\phi(t))(n) \neq h(x)(n)).$$

Let $Ct(2)$ be the total elements of X.

Corollary 1 *Let Φ be a closed formula with quantifiers over $\mathbb{N} \to \mathbb{N}$ and $\mathbb{N}$ and with a predicate variable Q.*
Recursivly in Φ we may find a $Ct(2)$-formula $\hat{\Phi}$ such that Φ is true for all β-sets if and only if $\hat{\Phi}$ has a $Ct(2)$-proof.

This is how far we will stretch the comparison with β-logic.

Π^1_n-logic In our next example we generalise this. Let Π_n be a complete Π^1_n-set, π_n the defining formula. Let Ψ_n be the statement

$$\forall g (\pi_n(g) \to g \in \Pi_n).$$

It is clear that a formula Φ is true for all sets A for which all Π^1_n-sets are absolute if and only if $\Psi_1 \wedge \ldots \wedge \Psi_n \to \Phi$ is true for all substructures of $(\mathbb{N} \to \mathbb{N}, \Pi_1, \ldots, \Pi_n)$.
The completenes theorem then implies that Φ is true in all these models if and only the transcript of $\Psi_1 \wedge \ldots \wedge \Psi_n \to \Phi$ has a computable, continuous proof.

E-logic In our final example we will develope what we call *E-logic*. If $A \subseteq \mathbb{N}^\mathbb{N}$ is a set, we may restrict the schemes for computation in ${}^3E$ to A. In this example we will accept ${}^3E$ as a hidden input, so when we write $\{e\}(\vec{f})$ or

$\{e\}^A(\vec{f})$ we mean a Kleene-computation with ${}^3E$ or ${}^3E$ restricted to A added as an input at the appropriate location (which will be coded in e). Also, when we write *computable* or *semicomputable* we will mean these concepts relativised to ${}^3E$.
We will let A be a subset of $\mathbb{N}^{\mathbb{N}}$ throughout this example.

Definition 17 A is an *E-structure* if we for all $\vec{f} \in A^k$ and all e, n have

$$\{e\}(\vec{f}) \approx n \Rightarrow \{e\}^A(\vec{f}) \approx n$$

The concept of an E-structure is co-semicomputable, so truth in all E-structures will be semicomputable.
E-structures may contain non-standard computations, i.e. computation tuples that is believed to be a convergent computation by the E-structure, but which is divergent in the full universe. It is easy to see that any Moschovakis witness of a non-standard computation will manifest itself only at the first non-reflecting ordinal. If R is a semicomputable set, we can construct a co-semicomputable extension R' as the set of functions believed to be in R by at least one E-structure. This is a better, and seems more natural, co-semicomputable approximation to R than the set with no Moschovakiswitness at stage κ_0 (The ordinal of lightface computation in ${}^3E$).

We now consider the domain where the total objects are the objects of the form $(e, \vec{f}, n)$ where e and n are natural numbers and each f in $\vec{f}$ is either a number or a function. We call such objects *computation tupples*. We let E be the set of valid computation tuples, i.e. those that represent a genuine computation in ${}^3E$. If A is a set, we let $\mathcal{A} = (A^+, E \cap A^+)$, where A^+ is the set of computation tuples where all functions come from A.
For simplicity we let 8_e be the index for the computation ${}^3E(\lambda g.\{e\}(g, \vec{f}))$.

Lemma 4 *A is an E-structure if and only if*

$$\mathcal{A} \models \forall e, \vec{f}((8_e, \vec{f}, 1) \in E \rightarrow \exists g((e, g, \vec{f}, 0) \notin E))$$

Proof
Using induction on the real computation, this proof is trivial.

Now let Θ be the statement

$$\Theta = \forall (e, \vec{f})((8_e, \vec{f}, 1) \in E \rightarrow \exists g((e, g, \vec{f}, 0) \notin E))$$

Let Φ be a first order statement over $\mathbb{N}^{\mathbb{N}}$ where the predicate E occurs positivly if it occurs at all. We have that Φ is true in all E-structures if and only if $\neg\Theta, \Phi$ is true in all structures $\mathcal{A}$.
For a further analysis, we will make use of the fact that the domain (S, S_{wf}) used for representing computations in ${}^3E$ in Normann [32] will satisfy co-density and thus accept h-functions. Moreover we will make use of the fact that we have a genuine reduction ϕ of the valid computation tuples to S_{wf}. Let $\breve{S} = \{s \in S \mid h(s) \text{ is total and } s \notin S_{\mathrm{wf}}\}$. Then $\breve{S}$ will be an alternative totality on S which will be co-semirecursive.
We see that E will only occur positivly in $\neg\Theta, \Phi$, so in the transcription to the language of a domain-based logic, we may replace terms $t \in E$ with $\forall s \in \breve{S}\exists n(h(s)(n) \neq h(\phi(t))(n))$. We then have that Φ is true in all E-structures if and only if the transcript of $\neg\Theta, \Phi$ has a continuous $\breve{S}$-proof.

Now it is important to notice that the continuous $\breve{S}$-proof is nothing more than an element in the domain of partial S-proofs relative to the language used. Thus it makes sense to ask if this proof is well-founded when we use an alternative subset of S as our branching-set. This will indeed be the case:

Definition 18 Let S_α be the elements of S_{wf} of rank $\leq \alpha$, and let $\breve{S}_\alpha$ be the complement of S_α with respect to the set of s with total $h(s)$.

Lemma 5 *Let P be a computable $\breve{S}$-proof. Then there is a computable ordinal α such that P is a well-founded $\breve{S}_\alpha$-proof.*

Proof:
Using the recursion theorem we design an algorithm on the total nodes of P essentially executing itself on subnodes in the proof tree and terminating trivially at axiom nodes. In the case of a subnode $F(s)$ of the $\breve{S}$-rule, where $h(s)$ is total, it will simultanously try to execute itself on the subnode and try to verify that $s \in S_{\mathrm{wf}}$. It is easy to see that this argorithm actually will terminate on P, and then with a computation of length α for some computable ordinal α. It is clear that P also is a well-founded $\breve{S}_\alpha$-proof.

We then have the following

Theorem 5 *Let Φ be a first order statement where the predicate E only occurs positivly.*
Φ is valid in all E-structures if and only if there is a computable $\breve{S}_\alpha$-proof for the transcript of $\neg\Theta, \Phi$.

We will give an application of this, using the set variable Q. We will let $I(Q)$ be a formula ensuring that an interpretation of Q in an E-structure A will contain all computation tuples converging in the sense of A. $I(Q)$ will have the format

$$\forall\tau(Sup(Q,\tau) \rightarrow \tau \in Q)$$

where Q is positive in Sup and τ is a sequence $(e, \vec{f}, n)$.

In our application we have to use a generalised version of E-proofs:

Definition 19 Let κ be an ordinal. A κ, E-proof is a proof obeying the rules of E-logic where applications of the the $\breve{S}_\kappa$-rule are replaced by the axioms $E(\tau)$ for computations τ of rank $\leq \kappa$, and where we in addition will acept axioms $Q(\tau)$ for the same computation-tuples τ

Lemma 6 *Let Q, E be positive in Δ_1 , ..., Δ_n and assume that P is an α_K-κ, E-proof without cut of*

$$\neg\Theta, \neg I(Q), \Delta_1, \ldots, \Delta_n.$$

Then uniformly computable in ${}^3E$, κ, α_K and P we can identify an index i and an ordinal $\kappa' \geq \kappa$ such that $\Delta_i(E, \mathcal{Q}_{\kappa'})$ is true, where $\mathcal{Q}_{\kappa'}$ is the set of computations of rank $\leq \kappa'$

Proof
We essentially use induction on the rank of P, and there will be several cases according to the last rule used in P.
Case 1 $\neg\Theta, \neg I(Q), \Delta_1, \ldots, \Delta_n$ is an axiom.
Then some Δ_i will be of the form $E(\tau)$ or $Q(\tau)$. Uniformly in κ we can identify one.
Case 2 One of the Δ_i's are introduced via the $\forall$, $\wedge$ or $\exists$-rule.
This case is trivial by the induction hypothesis.
Case 3 We introduce $\neg\Theta$, i.e. we deduce $\neg\Theta, \neg I(Q), \Delta_1, \ldots, \Delta_n$ from

$$\neg\Theta, \neg I(Q), ((8_e, \vec{f}, 1) \in E \wedge \forall g(e, g, \vec{f}, 0) \in E), \Delta_1, \ldots, \Delta_n$$

where e, $\vec{f}$ are given by terms. By the induction hypothesis we can identify one of these statements as true for $\mathcal{Q}_{\kappa'}$, and this cannot be $((8_e, \vec{f}, 1) \in E \wedge \forall g(e, g, \vec{f}, 0) \in E)$, since this is false. Thus we identify one of the Δ_i's

as true.
Case 4 We introduce $\neg I(Q)$, i.e. we deduce

$$\neg\Theta, \neg I(Q), (\neg Q(\tau) \wedge Sup(Q,\tau)), \Delta_1, \ldots, \Delta_n.$$

By if neccessary rewriting the proofs, we will have subproofs of

$$\neg\Theta, \neg I(Q), \neg Q(\tau), \Delta_1, \ldots, \Delta_n$$

and

$$\neg\Theta, \neg I(Q), Sup(Q,\tau), \Delta_1, \ldots, \Delta_n$$

and the induction hypothesis will apply to the seccond case. If we identify one of the Δ_i's as true, we are through. Otherwise we identify $Sup(Q,\tau)$ as true below an ordinal κ'. In that case we rewrite the proof of $\neg\Theta, \neg I(Q), \neg Q(\tau), \Delta_1, \ldots, \Delta_n$ to a proof of $\neg\Theta, \neg I(Q), \Delta_1, \ldots, \Delta_n$ by keeping all structure exept in the case of the rule of equality

$$\frac{\Gamma, \sigma = \tau}{\Gamma, Q(\sigma), \neg Q(\tau)}$$

In that case $Q_{\kappa'+1}(\sigma)$ will hold, and we simply plug in the axiom $\Gamma, Q(\sigma)$ instead.
We may then apply the induction hypothesis (which is by the ordinal hight of the proof and is uniform for all κ) to this reconstructed proof and will identify one of the Δ_i's as valid for some ordinal $\kappa'' \geq \kappa' + 1$. This ends the proof of the lemma.

We then get as a consequence

Theorem 6 *Let $\Delta(Q)$ be a positive statement such that $(\mathcal{A}, \mathcal{Q}) \models \Delta(Q)$ whenever $\mathcal{A}$ is an E-structure and $\mathcal{Q}$ is the set of terminating computations in the sense of $\mathcal{A}$.*
Then there is a computable ordinal κ such that $\Delta(\mathcal{Q}_\kappa)$ is true, where $\mathcal{Q}_\kappa$ is the set of computations terminating with ordinal rank $\leq \kappa$.

Corollary 2 *Let R be a semicomputable set. Then the following are equivalent:*

i) *R contains a non-empty computable subset*

ii) $A \models R \neq \emptyset$ *for all* E*-structures* A.

A further consequence will be that not every E-structure will have Moschovakis witnesses or alternative semicomputable sets of witnesses of non-termination. Slaman (unpublished) constructed a co-semicomputable admissible hull of any E-closure using the Kechrish basis theorem (see Sacks [39]), and in paricular he constructed an admissible E-structure. Slaman's construction can also be used as a basis for our corollary. It seems that we have replaced the use of a clever Skolem-hull argument by the general completenes proof.

References

[1] S. Abramsky and A. Jung, *Domain Theory*, Manuscript, Will appear in 'Handbook of logic in computer science'.

[2] K.J. Barwise, R.O. Gandy and Y.N. Moschovakis, *The next admissible set*, Jour. Symb. Log. 36 (1971)108-120

[3] U. Berger, *Totale Objekte und Mengen in der Bereichtheorie*, Thesis der Ludwig-Maximillians-Universität München (1990)

[4] U. Berger, *Total sets and objects in domain theory*, Annals of pure and applied logic 60 (1963) 91-117

[5] U. Berger, *Density for dependent types with totality*, Draft paper (1995)

[6] W. Buchholz and S.S. Wainer, *Provably computable functions and the fast growing hierarchy*, In S.G. Simpson *Logic and combinatorics* AMS Contemporary Mathematics Series 65 (1987) 179-198

[7] T. Engen, *Kvasi-vellfunderte trær og* ω*-logikk* (in Norwegian), Cand. Scient thesis, University of Oslo (1995)

[8] Yu.L. Ershov, *Computable functionals of finite type*, Algebra and Logic 11 (1972), 203-277

[9] Yu.L. Ershov, *The Theory of Numerations*, Vol 2 (in Russian), Novosibirsk (1973)

[10] Yu.L. Ershov, *Maximal and everywhere defined functionals*, Algebra and Logic 13 (1974), 210-225

[11] J.E. Fenstad, *General Recursion Theory. An Axiomatic Approach*, Springer Verlag (1980)

[12] R.O. Gandy, *Proof of Mostowski's conjecture*, Bull Acad. Polon. Sci. Math. 8 (1960) 571-575

[13] J.-Y. Girard, *Une extension de l'interprétation de Gödel à l'analyse, et son application à l'elimination des coupures dans l'anayse et la théorie des types*, in J.E. Fenstad (ed.) Proc. 2nd Scandinavian Logic Symposium, North-Holland (1971) 63-92

[14] J.-Y. Girard, *A survey of* Π^1_2*-logic*, L.J. Cohen, J. Łos, H. Pfeiffer and K.-P. Podewski (eds.) Proceedings of the international congress of logic, methodology and philosophy of science, North-Holland (1982) 89-107

[15] J.-Y. Girard, *Proof Theory and Logical Complexity Vol. I*, Bibliopolis (1987)

[16] J.-Y. Girard, *The system F of variable types fifteen years after*, Theor. Comp. Sci. 45 (1986) 159-192

[17] E.R. Griffor and M. Rathjen, *The Strength of Martin-Löf Type Theories with Well-Ordering Types*, Arch. Math. Logic 33 (1994), 347-385

[18] L. Harrington, *Contributions to recursion theory in higher types*, Ph.D. thesis MIT (1973)

[19] P. Hinman, *Recursion-Theoretic Hierarchies*, Springer (1978)

[20] S.C. Kleene, *Recursive functionals and quantifiers of finite types I*, T.A.M.S. 91 (1959) 1-52

[21] G. Kreisel, *On the interpretation of non-finitist proofs II*, J. Symb. Logic 17 (1952) 43-58

[22] G. Kreisel, *Interpretation of analysis by means of of functionals of finite types*, in A. Heyting (ed.) Constructivity in mathematics, North Holland (1959) 101-128

[23] L. Kristiansen, *Totality in Qualitative Domains*, Dr. Scient. Thesis, University of Oslo (1993)

[24] L. Kristiansen and D. Normann, *Semantics for some constructors of type theory*, in Behara, Fritsch and Lintz (eds.) Symposia Gaussiana, Conf. A, Walter deGruyter & Co (1995) 201-224

[25] L. Kristiansen and D. Normann, *Interpreting higher computations as types with totality*, Archive for Mathematical Logic 33 (1994) 243-259

[26] L. Kristiansen and D. Normann, *Total objects in inductivly defined types*, Oslo Preprint Series in Mathematics No. 4 (1995) To appear in Archive for Mathematical Logic

[27] P. Martin-Löf, *Intuionistic Type Theory*, Bibliopolis (1984)

[28] J. Moldestad, *Computations in Higher Types*, Springer Lecture Notes in Mathematics 574, Springer (1977)

[29] Y.N. Moschovakis, *Hyperanalytic Predicates*, T.A.M.S. 129 (1967) 249-282

[30] D. Normann, *Set recursion*, in J.E. Fenstad, R.O. Gandy and G.E. Sacks (eds.) Generalized Recursion Theory II, North-Holland (1978), 303-320

[31] D. Normann, *Formalizing the notion of total information*, in P.P. Petkov (ed.) Mathematical Logic, Plenum Press (1990) 67-94

[32] D. Normann, *Closing the gap between the continuous functionals and recursion in* ${}^3E$, to appear in the procedings of the Sacks conference MIT 1993, a special issue of Archives for Mathematical Logic.

[33] D. Normann, *A Hierarchy of Domains with Totality but without Density*, in Cooper, Slaman and Wainer (eds.) Computability, Enumerability, Unsolvability Directions in recursion theory, Cambridge University Press (1996) 233-257

[34] D. Normann, *Representation theorems for transfinite computability and definability*, in preparation

[35] D. Normann, *Categories of domains with totality*, in preparation

[36] E. Palmgren and V. Stoltenberg-Hansen, *Domain interpretations of Martin-Löfs partial type theory*, Annals of Pure and Applied Logic 48 (1990) 135-196

[37] M. Rathjen, *Admissible Proof Theory and Beyond*, in D. Prawitz, B. Skyrms and D. Westerstål (eds.) Proceedings of the 9th International Congress of Logic, Methodology and Philosophy of Science.

[38] H. Rogers Jr., *Theory of Recursive Functions and Effective Computability*, McGraw-Hill (1967)

[39] G.E. Sacks, *Higher recursion theory*, Springer (1990)

[40] D.S. Scott, *Continuous Lattices*, in E. Lawvere (ed.) Toposes, algebraic Geometry and Logic, Springer Lecture Notes in Mathematics 247 (1972) 97-136

[41] A. Setzer, *Proof theoretical strength of Martin-Löf Type Theory with W-type and one universe*, Thesis, der Ludwig-Maximillians-Universität München (1993)

[42] R. Soare, *Computability and Recursion*, to appear in the proceedings of the 10th International Congress of Logic, Methodology and Philosophy of Science, Firenze 1995

[43] C. Spector, *Hyperarithmetic quantifiers*, Fund. Math. 48 (1959) 313-320

[44] V. Stoltenberg-Hansen, I. Lindström and E.R. Griffor, *The mathematical theory of domains*, Cambridge University Press (1994)

[45] G. Waagbø, *Denotational Semantics for Intuitionistic Type Theory Using a Hierarchy of Domains with Totality*, Manuscript (1995)