Inductive Definitions with Decidable Atomic Formulas

Anton Setzer *

Department of Mathematics, Uppsala University, P. O. Box 480, S-75106 Uppsala, Sweden. email: setzer@math.uu.se

Abstract. We introduce a type theory for infinitely branching trees, called the theory of free algebras. In this type theory we define an extensional equality based on decidable atomic formulas only. We show, that equality axioms, which add full extensionality to the theory, yield a conservative extension of the (intensional) type theory for formulas having types of level ≤ 1. Types like nat $\to$ nat and well-founded trees with branching over the natural numbers (Kleene's O) have this property. We can therefore extract constructive proofs and programs from classical proofs of Π_2-sentences with this restriction on the types.

1 Introduction

In a series of articles [Be93], [Be95], [BS95a], [BS95b], [BS96], [Sch91], [Sch92], [Sch93a], [Sch93b], [Sch94], [Sch95] Schwichtenberg and Berger have studied the extraction of programs from classical proofs. This method relies on the fact, that in Π_1 assumptions, only decidable atomic formulas are used. When working in HA^ω we have no problems, since we can define an extensional equality easily based on decidable prime formulas.

Together with U. Berger, the author has introduced an extension of HA^ω by adding free algebras or well founded trees with branching over arbitrary types. This extension was necessary in order to treat term systems used for representing for instance non-deterministic terminating computation trees, infinitary term systems or, for implementations of proof theoretical techniques, infinitary formulas and proofs. The latter allows to extract programs from proof theoretical proofs and to work with infinitary logic in type theory. As expected, it turned out that some inductive definition can be defined directly in this theory and that we have the full proof theoretical strength of finitely iterated inductive definitions, as shown by the author ([Se94]).

To make use of this theory for the extraction of programs from classical proofs using equality, such an equality needs to be defined. Since we want to use

* Part of this research was supported by the EU-supported Twinning Project "Proofs and Computation", Leeds-Munich-Oslo. The research was written up, while the author, at that time based in Munich, was visiting the universities of Stockholm and Uppsala. The author wants to thank P. Martin-Löf for inviting him and making this fruitful visit possible and the logic group in Uppsala for providing such a creative and thoughtful environment.

such an equality in assumptions as well, we need an equality based on decidable prime formulas. In this article we show, that, such an equality can be defined in this extension and that we can prove it is extensional (however, we don't have all equality axioms available).

The definition of the equality is quite transparent, and uses ideas the author developed for well-ordering proofs in [Se93], [Se95]: We define the type of branches in a free algebra which is a list of indices, where each index potentially leads to a subtree of a term in a free algebra. If t is a term of the free algebra and b is a branch, then we define $t[b]$ to be the subtree reached following this branch to subtrees in t. Now t and t' are equal, if at every branch b $t[b]$ and $t'[b]$ are locally equal: the constructors are the same and all labels are the same: $t =_\sigma t' \leftrightarrow \forall x^{\mathrm{branch}(\sigma)}.\mathrm{loceq}_\sigma(t[x], t'[x])$. This definition is quite simple, and can be easily implemented.

However, the equality axioms can not be shown. This is no problem, since we can show, that adding such axioms yields a conservative extension of the original theory with respect to formulas using types of level ≤ 1 only.

The content of this article is as follows: In Sect. 2 we introduce the theory of free algebras FA. In order to make proofs easily, we restrict it to non simultaneous inductive definitions (and get the theory FA_1). The equality in FA_1 is introduced in Sect. 3 and we show, that we have full extensionality. In Sect. 4 we introduce equality axioms. We then show, that these equality axioms can be cut out: adding the equality axioms yields a conservative extension of FA_1 for formulas of type level ≤ 1. It follows, that adding to FA_1 an extensional equality yields a conservative extension for these formulas, and that we can extract constructive proofs and programs from classical proofs of Π_2-sentences of this level using the extensional equality defined in this article.

Related work. The treatment of strictly positive inductive definitions by introducing labeled trees is used in many type systems. For instance C. Paulin-Mohring has introduced similar inductive structures in the system Coq ([PM93]). The W-type of Martin-Löf (see his book [ML84]) is very close to the free algebras, and Petersson and Synek have extended it to labeled trees (very well described in [NPS90], Chap. 16). U. Berger has in [Be94] given a treatment of non strictly positive inductive definitions as well. The treatment of an equality based on atomic prime formulas in such type systems is to our knowledge new.

Elimination of extensionality was first studied by R. Gandy ([Ga56], [Ga59]). Afterwards H. Luckhardt improved these results ([Lu73]). A very good treatment of elimination of extensionality in HA^ω, written by A. Troelstra and based on ideas of W. Howard can be found in [Tr73], pp. 155 ff.

2 Definition of Free Algebras

The idea of free algebras of the strength in consideration is to generalize the principle of free algebras to trees with infinite branching.

The usual definition of a free algebra σ is, that we have several constructors C_i, each of which has some type $C_i : \alpha_1 \to \cdots \to \alpha_n \to \sigma^k \to \sigma$.

The semantics of this free algebra is the least set of closed terms, closed under these constructors.
Typical examples are:

- The type $\mathcal{B}$ of boolean truth values with constructors true : $\mathcal{B}$, false : $\mathcal{B}$.
- The type nat having constructors $0 : \mathrm{nat}$, $S : \mathrm{nat} \to \mathrm{nat}$.
- The type with k elements $\underline{\mathrm{n}}_k$, having constructors $i_k : \underline{\mathrm{n}}_k$ for $i < k$.
- If $\alpha_1, \ldots, \alpha_n$ are types, then $\alpha_1 \times \cdots \times \alpha_n$ is defined by one constructor $p_{\alpha_1,\ldots,\alpha_n} : \alpha_1 \to \cdots \to \alpha_n \to (\alpha_1 \times \cdots \times \alpha_n)$.
- $\alpha_1 + \cdots + \alpha_n$ is defined as the type having constructors $\mathrm{i}_{\alpha_1,\ldots,\alpha_n,i} : \alpha_i \to (\alpha_1 \times \cdots \times \alpha_n)$ $(i = 1, \ldots, n)$.
- If α is a type, the type $\mathrm{list}(\alpha)$ is defined by the constructors $\mathrm{Nil}_\alpha : \mathrm{list}(\alpha)$, $\mathrm{Cons}_\alpha : \alpha \to \mathrm{list}(\alpha) \to \mathrm{list}(\alpha)$.

(We will use these types later and omit the index α in Nil and Cons, and $\alpha_1, \ldots, \alpha_n$ in p and i).

In applications we often define several algebras simultaneously. For instance, we can introduce set terms and set formulas by the following definition: Variables are terms, if r, s are terms, then $\{r, s\}$ and $r \cup s$ are set terms, and if ϕ is a formula, a a set term, x a variable, then $\{x \in a \mid \phi\}$ is a set term. Formulas are $r \in s$ for r, s set terms, and if ϕ, ψ are set terms, x a variable, then $\phi \wedge \psi$, $\phi \vee \psi$ and $\forall x.\phi$ are formulas. This can be defined (if representing variables as natural numbers) as the free algebra with the following constructors:

- $\mathrm{Var} : \mathrm{nat} \to \mathrm{term}$.
- $\{\cdot\} : \mathrm{term} \to \mathrm{term} \to \mathrm{term}$.
- $\cup : \mathrm{term} \to \mathrm{term} \to \mathrm{term}$.
- $\mathrm{Separation} : \mathrm{nat} \to \mathrm{term} \to \mathrm{for} \to \mathrm{term}$.
- $\in : \mathrm{term} \to \mathrm{term} \to \mathrm{for}$.
- $\wedge, \vee : \mathrm{for} \to \mathrm{for} \to \mathrm{for}$.
- $\forall : \mathrm{nat} \to \mathrm{for} \to \mathrm{for}$.

This language has only limited expressiveness: nearly no comprehension is definable yet. In order to express at least restricted comprehension, we are immediately led to the idea of using infinitary formulas and terms:
If ϕ_n is a formula for $n : \mathrm{nat}$, then $\bigwedge_{n:\mathrm{nat}} \phi_n$ is a formula, and if a_n is a set term for $n : \mathrm{nat}$, then $\bigcup_{n:\mathrm{nat}} a_n$ is a set term, and and we need constructors:

- $\bigwedge : (\mathrm{nat} \to \mathrm{for}) \to \mathrm{for}$.
- $\bigcup : (\mathrm{nat} \to \mathrm{term}) \to \mathrm{term}$.

In order carry out some proof theoretical treatment of set theory in type theory, we need therefore free algebras with infinite branching.

In a theory using free algebras with infinitary branching, we can define for instance Kleene's O with the constructors $\mathrm{Nil} : \mathrm{O}$, $\mathrm{Succ} : \mathrm{O} \to \mathrm{O}$, $\mathrm{Lim} : (\mathrm{nat} \to \mathrm{O}) \to \mathrm{O}$.

We can define trees with finite or countable branching, by forming lists of trees l : treelist and defining from l a tree Succtree(l) having as subtrees the elements of the list l:

- Nil : treelist,
- $\text{Cons}_{\text{tree}}$: tree $\to$ treelist $\to$ treelist,
- $\text{Succ}_{\text{tree}}$: treelist $\to$ tree.
- Lim : (nat $\to$ tree) $\to$ tree.

The general concept of forming free algebras is, that, if we have constructors, where the type to be defined occurs strictly positive, than we get a new type. The semantics of this type in the term model is the least set of closed terms closed under these constructors. A rigorous definition of free algebras, which does not make use of the adding of new symbols is as follows:

Definition 1 of the type theory FA. (a) We inductively define the set of types of FA:
We first define the constructor types together with their free variables:
Let x_i for $i \in N$ be special symbols.
If $\alpha_i, \tau_{i,j}$ are types, $n \in \omega$, $m_i \in \omega$, then

$$\begin{aligned}&\alpha_1 \to \cdots \to \alpha_n \\ &\quad \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to x_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to x_{m_k}) \\ &\quad\quad \to x_l\end{aligned}$$

is a constructor type with free variables $\{x_{m_1}, \ldots, x_{m_k}, x_l\}$.
Now, if $\rho_1, \ldots, \rho_m$ are constructor types, $\text{FV}(\rho_j) \subset \{x_1, \ldots, x_n\}$, $1 \leq j \leq n$, then

$$\mu x_j.\mu x_1.\cdots\mu x_{j-1}.\mu x_{j+1}.\cdots\mu x_n.(\rho_1, \ldots, \rho_m)$$

is a type. We call this type a free algebra.
If σ, τ are types, then $(\sigma \to \tau)$ is a type.
We have the usual conventions about omitting parenthesis.

(b) We define for every free algebra its constructors together with the type of each constructor:
If $\sigma_j = \mu x_j.\mu x_1.\cdots\mu x_{j-1}.\mu x_{j+1}.\cdots\mu x_n.(\rho_1, \ldots, \rho_m)$, then $C(\sigma_1, \ldots, \sigma_n, i)$ $(1 \leq i \leq m)$ are the constructors of $\sigma_1, \ldots, \sigma_n$.
If

$$\begin{aligned}&\rho_i = \alpha_1 \to \cdots \to \alpha_n \\ &\quad \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to x_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to x_{m_k}) \\ &\quad\quad \to x_l \ ,\end{aligned}$$

then $C(\sigma_1, \ldots, \sigma_n, i)$ has type

$$\begin{aligned}&\alpha_1 \to \cdots \to \alpha_n \\ &\quad \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \sigma_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to \sigma_{m_k}) \\ &\quad\quad \to \sigma_l \ .\end{aligned}$$

(c) In the following we will introduce free algebras $(\sigma_j)_{1\le j\le n}$ by giving names for σ_j together with names for their constructors $(C_i)_{1\le i\le m}$ and by giving the types ρ_i of the constructors C_i. In this situation we say, that $(\sigma_j)_{1\le j\le n}$ are defined by the constructors $(C_i : \rho_i)_{1\le i\le m}$.

Functions can now be introduced by recursion over the definition of a free algebra. We could add the recursion operator. However, in a practical implementation in the proof assistant MINLOG, it turned out that one needs to add new functions which remain visible after normalization took place. Therefore we add functions defined by giving their step terms following the inductive definition of the free algebras. These functions will depend on terms.

Definition 2 of the terms of FA. (a) We define for every sequence of types $\vec{\gamma} = \gamma_1, \dots, \gamma_n$ and for simultaneously defined free algebras $\vec{\sigma} = \sigma_1, \dots, \sigma_n$ the step type for each constructor:
If σ_i is the free algebra, defined by the constructors C_i of type

$$\begin{aligned} C_i : {} & \alpha_1 \to \cdots \to \alpha_n \\ & \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \sigma_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to \sigma_{m_k}) \\ & \quad \to \sigma_l \ , \end{aligned}$$

the step type of C_i with respect to $\vec{\gamma}$ is

$$\begin{aligned} & \alpha_1 \to \cdots \to \alpha_n \\ & \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \sigma_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to \sigma_{m_k}) \\ & \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \gamma_{m_1}) \to \cdots \to (\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to \gamma_{m_k}) \\ & \quad \to \gamma_l \ . \end{aligned}$$

(b) Inductive definition of the set of terms together with their type (we write $t : \sigma$ or t^σ for t is a term of type σ):
Let for every type σ be infinitely many variables be given. We denote variables of type σ by $x^\sigma, y^\sigma, \dots$.
If x is a variable of type σ, then $x : \sigma$.
If $s : \sigma \to \rho$, $t : \sigma$ then $(st) : \rho$.
If $s : \rho$, x a variable of type σ, then $(\lambda x.s) : \sigma \to \rho$.
If C is a constructor of type α, $C : \alpha$.
If $\sigma_1, \dots, \sigma_n$ are free algebras simultaneously defined by the constructors $C_1, \dots, C_m$, $\vec{\rho} = \rho_1, \dots, \rho_n$ is a sequence of types, β_i is the step type of C_i with respect to $\vec{\rho}$, $t_i : \beta_i$, then $\mathrm{fun}_i(C_1 : t_1, \dots, C_m : t_m)$ is a term of type $\sigma_i \to \rho_i$.

(c) We will write $f(r_1, \dots, r_n)$ instead of $f\ r_1 \cdots r_n$, if $f : \gamma_1 \to \cdots \gamma_n \to \rho$. This does not cause any confusion with the product of terms, which is written as $\mathrm{p}r_1, \dots, r_n$ or in the new notation $\mathrm{p}(r_1, \dots, r_n)$.

(d) We define α-equality, β-conversion and η-conversion as usual.
Further we define the one-step R-conversion by:

If $f_i := \mathrm{fun}_i(C_1 : t_1, \ldots, C_m : t_m)$, and
C_i of the type as in (a), then

$$\begin{aligned} &f_l(C_i(l_1, \ldots, l_n, s_1, \ldots, s_k)) \to^1_R \\ &\quad t_i(l_1, \ldots, l_n, s_1, \ldots, s_k, \\ &\quad (\lambda x_1^{\tau_{1,1}}. \cdots \lambda x_{n_1}^{\tau_{1,n_1}}. f_{m_1}(s_1(x_1, \ldots, x_{n_1}))), \ldots, \\ &\qquad (\lambda x_1^{\tau_{k,1}}. \cdots \lambda x_{n_1}^{\tau_{k,n_k}}. f_{m_k}(s_k(x_1, \ldots, x_{n_k})))) \end{aligned}$$

for suitable variables x_i.
Further if $t \to^1_R t'$, then $s[x := t] \to^1_R s[x := t']$.
Let $\underline{\cong}$ is the reflexive, symmetric and transitive closure of α- β, η conversion and $\to_R$.
We will in the following identify $\cong$-equivalent terms.

(e) We will usually introduce functions by giving their convertibility-rules.

All the examples of types mentioned above can now be defined in FA.

Formulas can now be formed from atom(t) for $t : \mathcal{B}$ (so atom(true) is the verum and atom(false) is the falsum) and we get a logic built on induction over free algebras:

Definition 3 of the logic in FA. (a) We define the <u>formulas of FA</u>:
If $t : \mathcal{B}$, then atom(t) is an (atomic) formula.
If ϕ_1, ϕ_n are formulas, x^σ a variable, then $(\phi_1 \to \phi_2)$, $(\phi_1 \wedge \phi_2)$, $(\phi_1 \vee^* \phi_2)$, $(\forall x^\sigma.\phi_1)$, $(\exists^* x^\sigma.\phi_1)$ are formulas.
Note, that we identify $\cong$-equivalent terms.
We will introduce formulas in the form of $\phi(x)$ for some variable x. After having introduced it in this way, let $\phi(t) := \phi[x := t]$.

(b) $\underline{\bot} :=$ atom(false), $\underline{\neg\phi} := \phi \to \bot$.

(c) The <u>rules of FA</u> are the rules of minimal logic (intuitionistic logic without ex falsum quodlibet) with the strong exist quantifier $\exists^*$, strong disjunction $\vee^*$, and an axiom atom(true).
Additionally, we have the <u>axioms of induction</u> over free algebras:
Assume σ_i are free algebras, defined by the constructors $C_1 \cdots C_m$, C_i is of type as in Def. 2 (a) Then the <u>induction step of C_i with respect to $\phi_1(x^{\sigma_1}), \ldots, \phi_n(x^{\sigma_n})$</u> is

$$\begin{aligned} &\forall x_1^{\alpha_1}. \cdots \forall x_n^{\alpha_n}. \forall y_1^{\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \sigma_{m_1}}. \cdots \forall y_k^{\tau_{k,1} \to \cdots \to \tau_{k,n_k} \to \sigma_{m_k}}. \\ &\quad (\forall z_1^{\tau_{1,1}}. \cdots \forall z_{n_1}^{\tau_{1,n_1}}. \phi_{m_1}(y_1(z_1, \ldots, z_{n_1}))) \to \cdots \\ &\qquad \to (\forall z_1^{\tau_{k,1}}. \cdots \forall z_{n_k}^{\tau_{k,n_k}}. \phi_{m_k}(y_k(z_1, \ldots, z_{n_k}))) \\ &\qquad\quad \to \phi_l(C_i(x_1, \ldots, x_n, y_1, \ldots, y_k)) \ . \end{aligned}$$

If now $\psi_1, \ldots, \psi_m$ are the induction steps of $C_1, \ldots, C_m$ with respect to the formulas ϕ_i as above, then

$$\psi_1 \to \cdots \to \psi_m \to ((\forall x^{\sigma_1}.\phi_1(x^{\sigma_1})) \wedge \cdots \wedge (\forall x^{\sigma_n}.\phi_n(x^{\sigma_n})))$$

is an axiom of FA.

Proofs in FA are formed by the axioms using the rules. Each proof has a set of free assumptions and a set of free variables. (The free variables are the free (object-)variables in corresponding Curry-Howard terms, i.e. the free variables in the terms used for $\forall$-elimination and $\exists^*$-introduction, not bound later by $\forall$-introduction and $\exists^*$-elimination). Closed proofs are proofs that contain no free assumptions and only free variables occurring in the derived formula.[2]

Note, that we have ex falsum quodlibet and stability for atomic formulas by induction over $\mathcal{B}$, and therefore ex falsum quodlibet for all formulas, and stability for all formulas, which have no occurrence of $\vee^*$ or $\exists^*$.

We can simplify the free algebras by reducing simultaneous inductive definitions to non simultaneous ones. This is done by not distinguishing between the types defined before simultaneously:
If $\sigma_1, \ldots, \sigma_n$ are free algebras, defined by the constructors $C_1, \ldots, C_m$, let σ be a free algebra defined by the constructors $C'_1, \ldots, C'_m$, where, if C_i is of type as in Def. 2 (a),

$$\begin{aligned} C'_i : {} & \alpha_1 \to \cdots \to \alpha_k \\ & \to (\tau_{1,1} \to \cdots \to \tau_{1,n_1} \to \sigma) \to \cdots \to (\tau_{r,1} \to \cdots \to \tau_{r,n_r} \to \sigma) \to \sigma \ . \end{aligned}$$

We define $f_{\sigma_i,\sigma} : \sigma_i \to \sigma$ by

$$\begin{aligned} & f_{\sigma_i,\sigma}(C_i(l_1, \ldots, l_k, s_1, \ldots, s_r)) \cong \\ & \quad C'_i(l_1, \ldots, l_k, (\lambda x_1 . \cdots x_{n_1} . f_{\sigma_{m_1},\sigma}(s_1(x_1, \ldots, x_{n_1}))), \ldots, \\ & \qquad (\lambda x_1 . \cdots x_{n_r} . f_{\sigma_{m_r},\sigma}(s_1(x_1, \ldots, x_{n_r})))) \ . \end{aligned}$$

We can reduce the equality on σ_i to the equality on the type σ by applying $f_{\sigma_i,\sigma}$.

Further, we can replace $\tau_{i,1} \to \cdots \to \tau_{i,m_i} \to \sigma$ by $(\tau_{i,1} \times \cdots \times \tau_{i,m_i}) \to \sigma$, if $m_i \geq 1$, and define again an embedding. (We could reduce it even more, but this would lead to case distinctions, which are more complicated to deal with).

Therefore, it suffices to study in the following the following set of types:

Definition 4 of the type theory FA_1**.** (a) We define the set of types of FA_1 as a subset of FA:

[2] In the theories usually considered, every type is inhabited and we can replace free variables occuring in the proof and not occuring in free assumptions or the derived formula always by closed terms of the corresponding type, therefore the set of variables can be neglected. FA has empty types, and we have to take care of the free variables. Independently R. Stärk and the unknown referee – we would like to thank both for their comments – pointed out, that there are proofs of $\bot$ in FA: take the induction axiom $\forall x^{\underline{\mathrm{n}}_0} . \bot$, where $\underline{\mathrm{n}}_0$ is the empty type with no constructors, and use an $\forall$-elimination with the term $x^{\underline{\mathrm{n}}_0}$. However, this proof depends on the free variable $x^{\underline{\mathrm{n}}_0}$, is therefore not closed – the consistency of FA is not violated.

Assume σ is a free algebra, defined by constructors C_i, s.t. each constructor C_i has type

$$\begin{aligned}&\alpha^i_1 \to \cdots \to \alpha^i_{m_i}\\ &\quad \to (\tau^i_1 \to \sigma) \to \cdots \to (\tau^i_{n_i} \to \sigma) \to \sigma^{p_i} \to \sigma \ ,\end{aligned}$$

where $\sigma^{p_i} \to \sigma$ abbreviates $\underbrace{\sigma \to \cdots \to \sigma}_{p_i \text{times}} \to \sigma$ and α^i_j, τ^i_j are types of FA_1.

Then σ is a type of FA_1.

If σ, ρ are types of FA_1, then $(\sigma \to \rho)$ is a type of FA_1.

(b) If C_i has the type as in (a), the arguments of type α^i_j will be denoted by l_j (for *l*abel), of type $\tau^i_j \to \sigma$ by s_j and the remaining arguments by t_j, all possibly with accents etc. We denote the arguments by r_j, if we don't care about the distinction between the groups of arguments. Further let in the following, as arguments of C_i, $\vec{l} = (l_1, \ldots, l_{m_i})$, $\vec{s} = (s_1, \ldots, s_{n_i})$, $\vec{t} = (t_1, \ldots, t_{p_i})$, $\vec{l'} = (l'_1, \ldots, l'_{m_i})$, etc.

(c) Terms, formulas of FA_1 are the terms, formulas of FA which contain only FA_1-types. The rules and axioms of FA_1 are those of FA which use only FA_1-types.

(d) In the following, if not mentioned differently, C_i will have the type as denoted in Def. (a).

3 Definition of the Equality

As we noted in the introduction, the equality for free algebras is defined as: $t =_\sigma t'$ if $t[b]$ and $t'[b]$ are locally equal, for every branch b. Local equality means, that the last constructor is the same and the labels on $t[b]$ and $t'[b]$ are the same. Now the labels at $t[b]$, $t[b']$ depend on the constructor used. Since we have no dependent types, we need to extract a label, even, if the last constructor does not contain such a label. We deal with this problem by deciding, whether a type is empty or not. If it is empty, we do not need to extract a label of that type. If it is non empty, we have one fixed element, and use, in case the constructor contains no label, this fixed element (ϵ^σ) as the result of the label-extracting function.

Definition 5 of $\mathrm{E}^{\sigma\to\rho}$ and ϵ^ρ. For each type of FA_1 we define, whether it is empty or not, and define if σ is non empty, a closed term ϵ^σ and if σ is empty, for each type ρ a closed term $\mathrm{E}^{\sigma\to\rho}$ (E stands for ex falsum quodlibet), and a proof of $\forall x^\sigma.\phi(x)$ for any formula $\phi(x^\sigma)$:
Case σ is a free algebra, defined by constructors C_i:
If for one constructor C_i we have for all j α^i_j are non empty, τ^i_j are empty, $p_i = 0$, then σ is non empty. If i is minimal such that C_i is such a constructor, then

$$\epsilon^\sigma := C_i(\epsilon^{\alpha^i_1}, \ldots, \epsilon^{\alpha^i_{m_i}}, \mathrm{E}^{\tau^i_1\to\sigma}, \ldots, \mathrm{E}^{\tau^i_{n_i}\to\sigma}) \ .$$

Otherwise σ is empty, and we define $\mathrm{E}^{\sigma\to\rho}$ as the function $\sigma\to\rho$ such that if C_i has a type as above,

$$\mathrm{E}^{\sigma\to\rho}(C_i(\vec{l},\vec{s},\vec{t})\cong \begin{cases} \mathrm{E}^{\alpha^i_j\to\rho}(l_j) & \text{if } \alpha^i_j \text{ is empty, } j \text{ minimal,}\\ \mathrm{E}^{\sigma\to\rho}(s_j(\epsilon^{\tau^i_j})) & \text{otherwise, if } \tau^i_j \text{ is non empty, } j \text{ miminal,}\\ \mathrm{E}^{\sigma\to\rho}(t_1) & \text{otherwise.}\end{cases}$$

Case $\sigma\to\tau$:
If σ is empty, $\sigma\to\tau$ is non empty, $\epsilon^{\sigma\to\tau}:=\mathrm{E}^{\sigma\to\tau}$.
Otherwise, if τ non empty, then $\sigma\to\tau$ is non empty, $\epsilon^{\sigma\to\tau}:=\lambda x^\sigma.\epsilon^\tau$.
Otherwise $\sigma\to\tau$ is empty, $\mathrm{E}^{(\sigma\to\tau)\to\rho}:=\lambda x^{\sigma\to\tau}.\mathrm{E}^{\tau\to\rho}(x(\epsilon^\sigma))$.

The proofs of $\forall x^\sigma.\phi(x)$ for σ empty can be defined in a similar way as $\mathrm{E}^{\sigma\to\rho}$, replacing recursion by induction.

We define now the equality on the types. Free algebras, hereditarily built from free algebras with no branching over types, have decidable equality, which we can define as a boolean valued function by recursion over the type:

Definition 6 of the decidable equality. We define inductively the set of types σ with decidable equality together with $=^{\mathcal{B}}_\sigma\colon \sigma\to\sigma\to\mathcal{B}$ (usually written infix).
If σ is empty, then σ has decidable equality, $=^{\mathcal{B}}_\sigma:=\lambda x,y.\mathrm{true}$.
If σ is a non empty free algebra, defined by constructors C_i, such that, for each constructor C_i of the usual typing $n_i=0$ and for all j α^i_j has decidable equality, then σ has decidable equality, and we define $=^{\mathcal{B}}_\sigma$ by

$$C_i(\vec{r}) =^{\mathcal{B}}_\sigma C_j(\vec{r'})\cong \mathrm{false} \text{ if } i\neq j \ ,$$

and

$$\begin{aligned} &C_i(\vec{l},\vec{t}) =^{\mathcal{B}}_\sigma C_i(\vec{l'},\vec{t'})\\ &\quad\cong l_1 =^{\mathcal{B}}_{\alpha^i_1} l'_1 \wedge_{\mathcal{B}} \cdots \wedge_{\mathcal{B}} l_{m_i} =^{\mathcal{B}}_{\alpha^i_{m_i}} l'_{m_i} \wedge_{\mathcal{B}} t_1 =^{\mathcal{B}}_\sigma t'_1 \wedge_{\mathcal{B}} \cdots \wedge_{\mathcal{B}} t_{p_i} =^{\mathcal{B}}_\sigma t'_{p_i} \ . \end{aligned}$$

where $\wedge_{\mathcal{B}}$ is the boolean valued function corresponding to the logical connective $\wedge$.

The crucial case are free algebras with branching indexed over types. For these trees we define index(σ) as the union of all types, over which we can reach an immediate subtree. We can define for non empty types its immediate predecessor with respect to an element of index(σ). branch(σ) is the list of indices, and by induction over this list we reach every subtree:

Definition 7 of label, constr, pred, $r[b]$, index, branch(σ), ι, ν. Assume σ is defined by constructors $C_1,\ldots,C_m$ with the usual typing and non empty.

(a) If α^i_j is non empty $(1 \le i \le m, 1 \le j \le m_i)$ we define $\underline{\text{label}^\sigma_{i,j}} : \sigma \to \alpha^i_j$ as

$$\text{label}^\sigma_{i,j}(C_k(\vec{l},\vec{s},\vec{t})) \cong \begin{cases} l_j & \text{if } i = k \\ \epsilon^{\alpha^i_j} & \text{otherwise.} \end{cases}$$

(b) If $1 \le i \le m$, define $\underline{\text{constr}_i{}^\sigma} : \sigma \to \mathcal{B}$,

$$\text{constr}^\sigma_i(C_j(\vec{r})) \cong \begin{cases} \text{true} & \text{if } i = j \\ \text{false} & \text{otherwise} \end{cases} .$$

(c) Let $\underline{\text{index}(\sigma)}$ be the free algebra defined by constructors $\underline{\iota^\sigma_{i,j}} : \tau^i_j \to \text{index}(\sigma)$ $(1 \le i \le m,\ 1 \le j \le n_i)$, and $\underline{\nu^\sigma_{i,j}} : \text{index}(\sigma)$ $(1 \le i \le m,\ 1 \le j \le p_i)$, i.e. $\text{index}(\sigma)$ is the disjoint union of τ^i_j, and of a type with one element for $1 \le i \le m,\ 1 \le j \le p_i$.

(d) Define the function $\underline{\text{pred}^\sigma} : \sigma \to \text{index}(\sigma) \to \sigma$, assigning the immediate subtree of a tree t^σ, indexed by $l : \text{index}(\sigma)$ as follows: If $s := C_i(\vec{l},\vec{s},\vec{t})$, then

$$\text{pred}^\sigma(s, \iota_{i',j}(r)) \cong \begin{cases} s_j(r) & \text{if } i = i' \\ \epsilon^\sigma & \text{otherwise,} \end{cases}$$

$$\text{pred}^\sigma(s, \nu_{i',j}) \cong \begin{cases} t_j & \text{if } i = i' \\ \epsilon^\sigma & \text{otherwise.} \end{cases}$$

(e) Let $\underline{\text{branch}(\sigma)} := \text{list}(\text{index}(\sigma))$.

(f) Define $\underline{\cdot[\cdot]^\sigma} : \sigma \to \text{branch}(\sigma) \to \sigma$, written as $s[b]^\sigma$ for $\cdot[\cdot]^\sigma(s,b)$, such that

$$s[\text{Nil}]^\sigma \cong s$$
$$s[\text{Cons}(i,b)]^\sigma \cong \text{pred}_\sigma(s,i)[b] \ .$$

We will usually omit the superscript σ in all constructors and functions defined above.

Two trees are equal, if the subtrees reached using the same branch are locally equal. Locally equal means, that the constructors are the same and the labels are the same:

Definition 8 of the equality $\underline{=_\sigma}$ on the type σ. The definition is by recursion on definition of σ.

- Case σ has decidable equality. Then $s =_\sigma t := \text{atom}(s =^{\mathcal{B}}_\sigma t)$.
- Case σ is a free algebra with non decidable equality, defined by constructors C_i with the usual typing:
  - If $s : \sigma$, $t : \sigma$, then $\underline{\text{loceq}_\sigma(s,t)}$ (local equality) is the formula, which is the conjunction of $\overline{\text{label}^\sigma_{i,j}(s)} =_{\alpha^i_j} \text{label}^\sigma_{i,j}(t)$ for $1 \le i \le m$, $1 \le j \le m_i$, τ^i_j non empty and $\text{constr}_i(s) =_{\mathcal{B}} \text{constr}_i(t)$ $(1 \le i \le m)$.
  - $s =_\sigma t := \forall x^{\text{branch}(\sigma)}.\text{loceq}_\sigma(s[x], t[x])$.
- Case $\sigma = \rho \to \tau$, with non decidable equality: $s =_\sigma t := \forall x^\rho.s(x) =_\tau t(x)$.

Theorem 9. *We can prove the following in* FA_1*:*

(a) $=_\sigma$ *is reflexive, symmetric and transitive.*
(b) If σ *is defined by constructors* C_i*,* $j \neq l$*, then* $\neg(C_j(\vec{r}) =_\sigma C_l(\vec{s}))$*.*
(c) If σ *is defined by constructors* C_i*,* $r_i, r_i' : \sigma_i$*, then*

$$C_i(r_1, \ldots, r_k) =_\sigma C_i(r_1', \ldots, r_k') \leftrightarrow (r_1 =_{\sigma_1} r_1' \wedge \cdots \wedge r_k =_{\sigma_k} r_k') \ .$$

Proof. (a) Immediate by Meta-induction on the types.
(b) Trivial.
(c) The case, where σ has decidable equality is immediate. Non decidable case:
Let $s := C_i(\vec{r}) =: C_i(\vec{l}, \vec{s}, \vec{t})$, $s' := C_i(\vec{r'}) =: C_i(\vec{l'}, \vec{s'}, \vec{t'})$. Then
$\mathrm{loceq}(s[\mathrm{Nil}], s'[\mathrm{Nil}])$ iff $l_j = l_j'$ for $1 \leq j \leq m_i$.
If $u : \tau_j^i$, $x : \mathrm{branch}(\sigma)$, then

$$\mathrm{loceq}(s[\mathrm{Cons}(\iota_{i,j}(u), x)], s'[\mathrm{Cons}(\iota_{i,j}(u), x)]) \leftrightarrow \mathrm{loceq}((s_j(u))[x], (s_j'(u))[x])$$

for $k \neq i$,

$$\mathrm{loceq}(s[\mathrm{Cons}(\iota_{k,j}(u), x)], s'[\mathrm{Cons}(\iota_{k,j}(u), x)]) \leftrightarrow \mathrm{loceq}(\epsilon^\sigma[x], \epsilon^\sigma[x])$$

where the right side holds by reflexivity. We have similar statements for $s[\mathrm{Cons}(\nu_{k,j}, x)]$ and it follows

$$\begin{aligned} & \forall x^{\mathrm{branch}(\sigma)} \mathrm{loceq}(s[x], s'[x]) \\ \leftrightarrow & \bigwedge_{1 \leq j \leq m_i} l_j =_{\alpha_j^i} l_j' \wedge \bigwedge_{1 \leq j \leq n_i} \forall u^{\tau_j^i} \forall x^{\mathrm{branch}(\sigma)} \mathrm{loceq}(s_j(u)[x], s_j'(u)[x]) \wedge \\ & \bigwedge_{1 \leq j \leq p_i} \forall x^{\mathrm{branch}(\sigma)} \mathrm{loceq}(t_j[x], t_j'[x]) \\ \leftrightarrow & \vec{l} = \vec{l'} \wedge \vec{s} = \vec{s'} \wedge \vec{t} = \vec{t'} \ . \end{aligned}$$

4 Equality Axioms

In order to make use of our equality, we need equality axioms, which are not provable in FA_1 (or at least this should be the case by extending proofs for HA^ω). We show, that $\mathrm{FA}_1 - \mathrm{Eq}$ is conservative over FA_1 for formulas with no occurrences of types of level ≥ 2 (we could allow negative occurrences of $\forall x^\sigma$ and positive occurrences of $\exists^* x^\sigma$ for $level(\sigma) \geq 2$).

Definition 10 of $\mathrm{FA}_1 - \mathrm{Eq}$**.** Let $\underline{\mathrm{FA}_1 - \mathrm{Eq}}$ be defined as FA_1, but with the additional $\underline{\text{axioms for the extensional equality}}$:
For any types ρ, σ:

$$\forall x^\rho, y^\rho, z^{\rho \to \sigma} . x =_\rho y \to z(x) =_\sigma z(y) \ .$$

For proving our conservativity result, we will follow the approach we found in [Sch94], pp. 52 – 53 for the case of the typed lambda calculus and which again goes back to [Tr73], pp. 155 ff., and adapt it to the situation of free algebras.

The method used, is to define the formula $r \sim_\sigma s$ meaning: r and s are hereditarily extensional equal. If $\phi^\sim$ is the result of restricting a quantifier $\forall x$ and $\exists^* x$ to $x \sim x$, we can show, that if $\mathrm{FA}_1 - \mathrm{Eq} \vdash \phi$, then $\mathrm{FA}_1 \vdash \phi^\sim$. Now for formulas having the restriction to types of level ≤ 1, $\phi \leftrightarrow \phi^\sim$ is provable in FA_1 and we are done.

Definition 11 of $\sim$. We define $r \sim_\sigma s$ for every type σ, and $r, s : \sigma$ by recursion on the definition of the types:
If σ has decidable equality, then $r \sim_\sigma s := r =_\sigma s$.
Assume now σ has a non decidable equality.
If σ is a free algebra defined by constructors C_i with the usual typing, then let $\mathrm{loceq}^\sim_\sigma(s,t)$ be the conjunction of $\mathrm{label}^\sigma_{i,j}(s) \sim_{\alpha^i_j} \mathrm{label}^\sigma_{i,j}(t)$ for $1 \leq i \leq m$, $1 \leq j \leq m_i$, τ^i_j non empty and $\mathrm{constr}_i(s) \sim_{\mathcal{B}} \mathrm{constr}_i(t)$ $(1 \leq i \leq m)$.

$$s \sim_\sigma t := \forall x^{\mathrm{branch}(\sigma)}, y^{\mathrm{branch}(\sigma)}.x \sim_{\mathrm{branch}(\sigma)} y \to \mathrm{loceq}^\sim_\sigma(s[x], t[y]) \ .$$

If $\sigma = \rho \to \tau$, then

$$t \sim_\sigma t' := (\forall x^\rho, y^\rho.x \sim_\rho y \to tx \sim_\tau t'y) \ .$$

Remark. More precisely, in the above definition we have to proceed as follows, since $\mathrm{branch}(\sigma)$ is in general more complex than σ:
$\mathrm{index}(\mathrm{index}(\sigma))$ has no constructors, therefore decidable equality, therefore as well $\mathrm{branch}(\mathrm{index}(\sigma)) = \mathrm{list}(\mathrm{index}(\mathrm{index}(\sigma)))$, and then we can define $\sim_{\mathrm{branch}(\mathrm{index}(\sigma))}$ and, using $\sim_{\tau_{i,j}}$, $\sim_{\mathrm{index}(\sigma)}$.
$\mathrm{index}(\mathrm{branch}(\sigma))$ has only constructors $\nu^{branch(\sigma)}_{i',j'} : \mathrm{index}(\mathrm{branch}(\sigma))$, therefore decidable equality, therefore as well $\mathrm{branch}(\mathrm{branch}(\sigma))$, and we get a definition of $\sim_{\mathrm{branch}(\mathrm{branch}(\sigma))}$ and, using $\sim_{\mathrm{index}(\mathrm{branch}(\sigma))}$ as well of $\sim_{\mathrm{branch}(\sigma)}$. Now we can define $\sim_\sigma$.

Lemma 12. *$\sim_\sigma$ is symmetric and transitive.*[3]

Proof. easy.

Definition 13 of level of types, N- and P-formulas. (a) We define the level of types $\mathrm{level}(\sigma)$ by recursion on the inductive definition of the type σ:
If σ has decidable equality, then $\mathrm{level}(\sigma) := 0$.
If σ is a free algebra with non decidable equality, defined by the constructors C_i, typed as usual, then

$$\begin{aligned}\mathrm{level}(\sigma) := \max\{&\mathrm{level}(\alpha^i_j) \mid 1 \leq i \leq m, 1 \leq j \leq m_i\} \\ &\cup\{\mathrm{level}(\tau^i_j) + 1 \mid 1 \leq i \leq m, 1 \leq j \leq n_i\} \ .\end{aligned}$$

[3] Note that reflexivity of $\sim_\sigma$ is not provable. By symmetry and transitivity follows $x \sim_\sigma y \to x \sim_\sigma x$

If $\sigma = \rho \to \tau$ with non decidable equality, then

$$\mathrm{level}(\sigma) := \max\{\mathrm{level}(\rho)+1, \mathrm{level}(\tau)\} \ .$$

(b) We define the P- and N-formulas:
atom(r) is a P and an N-formula.
If ϕ, ψ are P (N-)formulas, then $\phi \wedge \psi$, $\phi \vee^* \psi$ are P- (N-)formulas.
If ϕ is a P- (N-), ψ is an N- (P-)formula, then $\psi \to \phi$ is a P- (N-)formula.
If ϕ is a P-formula then $\exists^* x^\sigma \phi$ is a P-formula, and, if level$(\sigma) \leq 1$, $\forall x^\sigma.\phi$ is a P-formula.
If ϕ is an N-formula then $\forall x^\sigma.\phi$ is an N-formula, and, if level$(\sigma) \leq 1$, $\exists^* x^\sigma.\phi$ is an N-formula.

Lemma 14. *If σ has decidable equality, τ is a type, then*
$\forall x^\sigma, y^\sigma.x =_\sigma y \to \forall f^{\sigma\to\tau}.((\forall z^\sigma.f(z) \sim_\tau f(z)) \to f(x) \sim_\tau f(y))$.

Proof by Meta-induction on the definition of the types. If σ is empty, this follows immediately.
If σ is a free algebra, defined the constructors C_i with usual typing, α^i_j have decidable equality, $n_i = 0$, we show the assertion by induction on x^σ, side-induction on y^σ.
Assume $x \equiv C_i(\vec{l}, \vec{t})$, $l_i =_{\alpha^i_j} l'_i$, $t_i =_\sigma t'_i$ and assume the assertion for t_i.
Let

$$g_j := \lambda x_j^{\alpha^i_j}.f(C_i(l'_1, \ldots, l'_{j-1}, x_j, l_{j+1}, \ldots, l_{m_i}, t_1, \ldots, t_{p_i})) \ ,$$
$$h_j := \lambda x_j^{\sigma}.f(C_i(l'_1, \ldots, l'_{m_i}, t'_1, \ldots, t'_{j-1}, x_j, t_{j+1}, \ldots, t_{p_i})) \ .$$

By Meta-IH we have $g_j(l_j) \sim_\tau g_j(l'_j)$ and by IH for t_j, $h_j(t_j) \sim_\tau h_j(t'_j)$. By the transitivity of $\sim$ follows the assertion.

It causes slight technical problems that $r \sim_\sigma s$ depends on $r \sim_{\mathrm{branch}(\sigma)} s$, where branch$(\sigma)$ is in general more complicated than σ. In the following lemma we overcome this difficulty:

Lemma 15. *(a) Assume σ with non decidable equality, σ be defined by constructors C_i with the usual typing, $\epsilon^{\alpha^i_j} \sim_{\alpha^i_j} \epsilon^{\alpha^i_j}$, if α^i_j is non empty. Then* $\forall x^{\mathrm{branch}(\sigma)}.\mathrm{loceq}^\sim_\sigma(\epsilon^\sigma[x], \epsilon^\sigma)$.
(b) For all non empty types σ, $\epsilon^\sigma \sim \epsilon^\sigma$, and for all empty types σ and all types ρ, $\mathrm{E}^{\sigma\to\rho} \sim \mathrm{E}^{\sigma\to\rho}$.
(c) If branch(σ) *has decidable equality, then*
$r \sim_\sigma r' \leftrightarrow \forall x^{\mathrm{branch}(\sigma)}.\mathrm{loceq}^\sim_\sigma(r[x], r'[x])$.
(d) Assume σ defined by constructors C_i with usual typing. Then

$$\iota_{i,j}(r) \not\sim_{\mathrm{index}(\sigma)} \iota_{i',j'}(r') \ , \ \text{if } i \neq i' \vee j \neq j',$$
$$\iota_{i,j}(r) \sim_{\mathrm{index}(\sigma)} \iota_{i,j}(r') \leftrightarrow r \sim_{\tau^i_j} r' \ ,$$
$$\nu_{i,j} \not\sim_{\mathrm{index}(\sigma)} \nu_{i',j'} \ , \ \text{if } i \neq i' \vee j \neq j'.$$
$$\nu_{i,j} \sim_{\mathrm{index}(\sigma)} \nu_{i,j} \ .$$

(e) Assume σ defined by constructors C_i with usual typing. Then

$$\mathrm{Nil} \sim_{\mathrm{branch}(\sigma)} \mathrm{Nil}\ , \mathrm{Nil} \not\sim_{\mathrm{branch}(\sigma)} \mathrm{Cons}(i,b)\ ,$$
$$\mathrm{Cons}(i,b) \sim_{\mathrm{branch}(\sigma)} \mathrm{Cons}(i',b') \leftrightarrow i \sim_{\mathrm{index}(\sigma)} i' \wedge b \sim_{\mathrm{branch}(\sigma)} b'\ .$$

Proof. (a) $\epsilon^\sigma = C_i(\epsilon^{\alpha^i_1}, \ldots, \epsilon^{\alpha^i_{m_i}}, \mathrm{E}^{\tau^i_1 \to \sigma}, \ldots, \mathrm{E}^{\tau^i_{n_i} \to \sigma})$, α^i_j non empty, τ^i_j empty. Induction on $x^{\mathrm{branch}(\sigma)}$.
If $x \equiv \mathrm{Nil}$, the assertion is trivial.
If $x \equiv \mathrm{Cons}(i,b)$, then if $i \equiv \iota^i_j(s)$ with $s : \tau^i_j$, then by τ^i_j empty we get the assertion. Otherwise $\epsilon[x] \cong \epsilon[b]$ and the assertion follows by IH.

(b) $\mathrm{E}^{\sigma\to\tau} \sim \mathrm{E}^{\sigma\to\tau}$ follows from the emptiness of σ.
Proof of $\epsilon_\sigma \sim \epsilon_\sigma$ by Meta-induction on the definition of types:
If σ has decidable equality, the assertion follows by reflexivity of $=_\sigma$.
If σ is a free algebra, the assertion follows by the symmetry and transitivity of $\mathrm{loceq}^\sim_\sigma$ and (a).
If $\sigma = \rho \to \tau$, ρ is empty, $\epsilon^\sigma \equiv \mathrm{E}^{\rho\to\tau} \sim \mathrm{E}^{\rho\to\tau} \equiv \epsilon^\sigma$, and if ρ is non empty, τ is non empty, follows for all $x, y : \rho$, $\epsilon^\sigma(x) \cong \epsilon^\tau \sim \epsilon^\tau \cong \epsilon^\sigma(y)$.
(c) "$\to$" follows by the reflexivity of $\sim_{\mathrm{branch}(\sigma)} \equiv =_{\mathrm{branch}(\sigma)}$.
"$\leftarrow$": Assume $x \sim y$, i.e. $x = y$, $\forall x^{\mathrm{branch}(\sigma)}.\mathrm{loceq}^\sim_\sigma(r[x], r'[x])$. Then

$$\forall x^{\mathrm{branch}(\sigma)}.\mathrm{label}_{i,j}(r[x]) \sim \mathrm{label}_{i,j}(r'[x])\ ,$$

by Lemma 14

$$\mathrm{label}_{i,j}(r[x]) \sim \mathrm{label}_{i,j}(r'[y])\ .$$

Similarly $\mathrm{constr}_i(r[x]) \sim \mathrm{constr}_i(r'[y])$, $\mathrm{loceq}^\sim_\sigma(r[x], r'[y])$.
(d) and (e) follow using (b) and (c), since the types $\mathrm{branch}(\mathrm{index}(\sigma))$ and $\mathrm{branch}(\mathrm{branch}(\sigma))$ have decidable equality.

Now we are ready, to characterize $\sim_\sigma$ similarly to the characterization of $=_\sigma$ before:

Lemma 16. *Assume σ is a free algebra defined by constructors C_i with usual typing.*

(a) If $i \neq j$, then $C_i(\vec{r}) \not\sim_\sigma C_j(\vec{r}')$.
(b) Let $r := C_i(\vec{l}, \vec{s}, \vec{t})$, $r' := C_i(\vec{l}', \vec{s}', \vec{t}')$. Then

$$r \sim_\sigma r' \leftrightarrow \bigwedge_{j=1}^{m_i} l_j \sim l'_j \wedge \bigwedge_{j=1}^{n_i} s_j \sim s'_j \wedge \bigwedge_{j=1}^{p_i} t_j \sim t'_j\ .$$

Proof. If σ has decidable equality the assertions are easy.
(a): easy.
(b): "$\to$": $\mathrm{Nil} \sim \mathrm{Nil}$, $\mathrm{loceq}^\sim(r[\mathrm{Nil}], r'[\mathrm{Nil}])$, $l_i \sim l_i$. Further, if $b \sim_{\mathrm{branch}(\sigma)} b'$, $y \sim_{\tau^i_j} y'$, then $\mathrm{Cons}(\iota_{i,j}(y), b) \sim \mathrm{Cons}(\iota_{i,j}(y'), b')$, $r[\mathrm{Cons}(\iota_{i,j}(y), b)] \cong (s_j(y))[b]$, $\mathrm{loceq}^\sim(r[\mathrm{Cons}(\iota_{i,j}(y), b)], r'[\mathrm{Cons}(\iota_{i,j}(y'), b')])$, $\mathrm{loceq}^\sim(s_j(y)[b], s'_j(y')[b'])$, $s_j(y) \sim s'_j(y')$, $s_j \sim s'_j$, similarly follows $t_j \sim t'_j$.
"$\leftarrow$": By induction on $\mathrm{branch}(\sigma)$ we show $x \sim y \to \mathrm{loceq}^\sim(r[x], r'[y])$.

Lemma 17. *If* $\mathrm{level}(\sigma) \leq 1$, *then* $\forall x^\sigma . x \sim x$.

Proof by Meta-induction on the types. If σ has decidable equality, this is trivial. If σ is a free algebra defined by constructors C_i with usual typing, we use induction on $x : \sigma$, and have in case $x \equiv C_i(l_1, \ldots, l_{m_i}, s_1, \ldots, s_{n_i}, t_1, \ldots, t_{p_i})$, $l_i \sim l_i$ by Meta-IH, by IH $\forall x^{\tau_j^i} . s_j(x) \sim s_j(x)$, therefore by Lemma 14 $s_j \sim s_j$, by IH $t_j \sim t_j$, by Lemma 16 (b) $x \sim x$.
If $\sigma = \rho \to \tau$ follows for $x : \sigma$, by IH $\forall y^\rho . x(y) \sim x(y)$ and by Lemma 14 $x \sim x$.

Lemma 18. *If* $r(\vec{x}) := r[\vec{z} := \vec{x}]$, $\mathrm{FV}(r) \subset \{z_1, \ldots, z_n\}$.
Then $\forall \vec{x}, \vec{y} . \vec{x} \sim \vec{y} \to r(\vec{x}) \sim r(\vec{y})$.

Proof by Meta-induction on the definition of the terms. Case $r = C_i$ constructor: by Lemma 16 (b).
Case $r = f^{\sigma \to \tau}$ function defined from step terms r_i.
Assume $\vec{x}, \vec{y}, \vec{x} \sim \vec{y}$ and let $g := f[\vec{z} := \vec{x}]$, $g' := f[\vec{z} := \vec{y}]$, $\tilde{t}_i := r_i[\vec{z} := \vec{x}]$, $\tilde{t}'_i := r_i[\vec{z} := \vec{y}]$.
We show $\forall x^\sigma, y^\sigma . x \sim y \to g(x) \sim g'(y)$ by induction on $x, y : \sigma$.
Case $x \equiv C_i(\vec{l}, \vec{s}, \vec{t})$, $y \equiv C_i(\vec{l}', \vec{s}', \vec{t}')$.

$$g(x) \cong \tilde{t}_i(\vec{l}, \vec{s}, \vec{t}, (\lambda x . g(s_1(x))), \ldots, (\lambda x . g(s_{n_i}(x))), g(t_1), \ldots, g(t_{p_i})) \ .$$

$l_i \sim l'_i$, $s_i \sim s'_i$, $t_i \sim t'_i$, and if $u \sim_{\tau_j^i} u'$, then $s_j(u) \sim s'_j(u')$, by IH $g(s_j(u)) \sim g'(s'_j(u'))$, therefore $\lambda x . g(s_j(x)) \sim \lambda x . g'(s'_j(x))$, further by IH $g(t_j) \sim g'(t'_j)$, by Meta-IH $\tilde{t}_i \sim \tilde{t}'_i$, $g(x) \sim g'(y)$.
Case r an application or λ-abstraction: easy.

Definition 19 of $\phi^\sim$ and $\sim'_\sigma$. (a) Let for ϕ a formula, $\underline{\phi^\sim}$ be the result of binding all quantifiers $\forall$, $\exists^*$ by $\sim$: $(\forall x . \phi)^\sim := \forall x . x \sim x \to \phi^\sim$, $(\exists^* x . \phi)^\sim := \exists^* x . x \sim x \wedge \phi^\sim$, and for other formula constructions $\phi^\sim$ is the result of applying this operation to its components.
(b) $\underline{r \sim'_\sigma s} := (r =_\sigma s)^\sim$.

Lemma 20. *(a) Lemmata 15 and 16 follow with* $\sim$ *replaced by* $\sim'$, $\mathrm{loceq}^\sim_\sigma(r, s)$ *replaced by* $(\mathrm{loceq}_\sigma(r, s))^\sim$.
(b) $\forall x^\sigma, y^\sigma . x \sim_\sigma x \to y \sim_\sigma y \to (x \sim_\sigma y \leftrightarrow (x =_\sigma y)^\sim)$.
(c) $(\forall x, y, z . x =_\sigma y \to (z(x) =_\tau z(y)))^\sim$.

Proof. (a) easy.
(b) Meta-induction on the definition of the types. If σ has decidable equality, the assertion is trivial. If σ is a free algebra defined by constructors C_i, show the assertion by induction on $x, y : \sigma$. If $x \equiv C_i(r_1, \ldots, r_n)$, $y \equiv C_i(r'_1, \ldots, r'_n)$, $x \sim x$, $y \sim y$, then $r_i \sim r_i$, $r'_i \sim r'_i$,

$$x \sim_\rho y \leftrightarrow \bigwedge_i r_i \sim r'_i \leftrightarrow \bigwedge_i r_i \sim' r'_i \leftrightarrow x \sim' y$$

using Meta-IH and IH.
If $\sigma = \rho \to \tau$ the assertion is easy.
(c): by (b).

Theorem 21. *If* $\mathrm{FA}_1 - \mathrm{Eq} \vdash \phi$, $\mathrm{FV}(\phi) \subset \vec{x}$, *then* $\mathrm{FA}_1 \vdash \forall \vec{x}.\vec{x} \sim \vec{x} \to \phi^\sim$.

Proof by Meta-induction on the derivation. The logical rules are easy (for the rules $\forall$-elimination and $\exists^*$-introduction note that by lemma 18 for the term r used we have $\forall \vec{x}.\vec{x} \sim \vec{x} \to r \sim r$). The extensionality axiom was just treated. For every induction axiom ψ of a formula $\phi(x)$ we have $\psi^\sim$, since using Lemma 16 (b), from the premise of $\psi^\sim$ we can derive the step terms for the induction axiom for $\phi'(x) := x \sim x \to \phi^\sim(x)$.

Theorem 22. *(a)* $\mathrm{FA}_1 - \mathrm{Eq}$ *is a conservative extension for* FA_1 *for P-formulas: If* ϕ *is a closed P-formula,* $\mathrm{FA}_1 - \mathrm{Eq} \vdash \phi$, *then* $\mathrm{FA}_1 \vdash \phi$.
(b) Adding to FA_1 *an extensional equality yields a conservative extension of* FA_1 *for P-formulas*
(c) Especially this holds for formulas containing only types of level ≤ 1.

Proof. From $\mathrm{FA}_1 - \mathrm{Eq} \vdash \phi$ follows $\mathrm{FA}_1 \vdash \phi^\sim$. Now using Lemma 17 follows, if ψ is an N-formula, $\mathrm{FV}(\psi) \subset \{\vec{x}\}$, $\mathrm{FA}_1 \vdash \forall \vec{x}.\vec{x} \sim \vec{x} \to \psi \to \psi^\sim$ and if ψ is a P-formula, then $\mathrm{FA}_1 \vdash \forall \vec{x}.\vec{x} \sim \vec{x} \to \psi^\sim \to \psi$. Therefore we get $\mathrm{FA}_1 \vdash \phi$.
(b): by (a), since we can interpret FA_1 extended by axioms for an extensional equality in $\mathrm{FA}_1 + \mathrm{Eq}$.

References

[Be93] Berger, U.: *Program extraction from normalization proofs.* In: M. Bezem, J.F. Groote (Eds.): *Typed Lambda Calculi and Applications.* Springer Lecture Notes in Computer Science 664, 1993, pp. 91 – 106.

[Be94] Berger, U.: *A constructive interpretation of inductive definitions.* Draft, Dept. of Mathematics, University of Munich, 1994.

[Be95] Berger, U.: *Programs from classical proofs.* In: Behara, M., Fritsch, R.,Lintz, R. G. (Eds.): *Symposia Gaussiana. Proceedings of the 2nd Gauss Symposium. Conference A: Mathematics and Theoretical Physics*, Walter de Gruyter, Berlin, 1995, pp. 187 – 200.

[BS95a] Berger, U., Schwichtenberg, H.: *Program extraction from classical proofs.* In: Leivant, D. (Ed.): *Logic and Computational Complexity*, Springer Lecture Notes in Computer Science 960, 1995, pp. 77 – 97.

[BS95b] Berger, U., Schwichtenberg, H.: *Program development by proof transformation*, In: Schwichtenberg, H. (Ed.): *Proof and Computation. NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 20 – August 1, 1993.* Springer, Heidelberg, 1995, pp. 1 – 45.

[BS96] Berger, U., Schwichtenberg, H.: *The greatest common divisor: a case study for program extraction from classical proofs.* To appear in: *Proceedings of Proofs and Types, Turin 1995*, 1996.

[Ga56] Gandy, R.: *On the axiom of extensionality – part I, Journal of Symbolic Logic*, 21, 1956, pp. 36 – 48.

[Ga59] Gandy, R.: *On the axiom of extensionality – part II, Journal of Symbolic Logic*, 24, 1959, pp. 287 – 300.

[Lu73] Luckhardt, H.: *Extensional Gödel Functional Interpretation. A Consistency Proof of Classical Analysis.* Springer Lecture Notes in Mathematics 306, 1973.

[ML84] Martin-Löf, P.: *Intuitionistic type theory*, Bibliopolis, Naples, 1984.

[Mu90] Murthy, C.: *Extracting constructive content from classical proofs.* PhD thesis. Technical Report 90-1151, Dept. of Computer Science, Cornell University, Ithaca, New York, 1990.

[NPS90] Nordström, B., Petersson, K., Smith, J.: *Programming in Martin-Löf's type theory. An Introduction*, Oxford University Press, Oxford, 1990.

[PM93] Paulin-Mohring, C.: *Inductive definitions in the system Coq. Rules and Properties.* In: Bezem, M., Groote, J.F. (Eds.): *Typed lambda calculi and applications*, Springer Lecture Notes in Computer Science 664, 1993, pp. 328 – 345.

[Sch91] Schwichtenberg, H.: *Normalization.* In: Bauer, F. L. (Ed.): *Logic, Algebra and Computation.* Springer, Heidelberg, 1991, pp. 201 – 237.

[Sch92] Schwichtenberg, H.: *Minimal from classical proofs.* In: Börger, E. , Jäger, G., Kleine–Büning, H. , Richter, M. M.: *Computer Science Logic*, Springer Lecture Notes in Computer Science 626, 1992, pp. 326 – 328.

[Sch93a] Schwichtenberg, H.: *Minimal Logic for computable functions.* In: Bauer, F. L., Brauer, W., Schwichtenberg, H.: *Logic and Algebra of Specification, NATO Advanced Study Institute, International Summer School held in Marktoberdorf, Germany, July 23–August 4, 1991*, Springer, Heidelberg, 1993, pp. 289 – 320.

[Sch93b] Schwichtenberg, H.: *Proofs as Programs.* In: Aczel, P.,Simmons, H., Wainer, S. S.: *Proof Theory. A selection of papers from the Leeds Proof Theory Programme 1990.* Cambridge University Press, 1993, pp. 81 – 113.

[Sch94] Schwichtenberg, H.: *Proof Theory.* Manuskript, Dept. of Mathematics, University of Munich, 1994. Available via http://www.mathematik.uni-muenchen.de/~schwicht/lectures/proofth/ss94/pt.dvi.Z.

[Sch95] Schwichtenberg, H.: *Computational Content of Proofs.* To appear in *Proceedings of the Summerschool in Marktoberdorf*, 1995.

[Se93] Setzer, A.: *Proof Theoretical Strength of Martin–Löf Type Theory with W–Type and One Universe*, PhD thesis, University of Munich, Dep. of Mathematics, 1993.

[Se94] Setzer, A.: *A type theory for iterated inductive definitions*, Draft, Munich, 1994. Available via http://www.mathematik.uni-muenchen.de/~setzer/articles.

[Se95] Setzer, A.: *Well-ordering proofs for Martin-Löf Type Theory with W-type and one universe.* Submitted. A preliminary version is available via http://www.mathematik.uni-muenchen.de/~setzer/articles.

[TD88] Troelstra, A. S., Dalen, D. v.: *Constructivism in Mathematics. An introduction*, volume 121, 123 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam, 1988

[Tr73] Troelstra, A. S. (Ed.): *Metamathematical Investigation of Intuitionistic Arithmetic and Analysis.* Springer Lecture Notes in Mathematics 344, 1973.

This article was processed using the LaTeX macro package with LLNCS style