Available online at www.sciencedirect.com
SCIENCE DIRECT
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 122 (2005) 105–126
www.elsevier.com/locate/entcs

Reflection Into Models of Finite Decidable FP-sketches in an Arithmetic Universe

Maria Emilia Maietti

Dipartimento di Matematica Pura ed Applicata
University of Padova
via Belzoni n.7, 35100 Padova, Italy
maietti@math.unipd.it

Abstract

We consider finite decidable FP-sketches within an arithmetic universe. By an FP-sketch we mean a sketch with terminal and binary product cones. By an arithmetic universe we mean a list-arithmetic pretopos, which is the general categorical definition we give to the concept of arithmetic universe introduced by Andrè Joyal to prove Gödel incompleteness theorems.
Then, for finite decidable FP-sketches we prove a constructive version of Ehresmann-Kennison's theorem stating that the category of models of finite decidable FP-sketches in an arithmetic universe is reflective in the corresponding category of graph morphisms.
The proof is done by employing the internal dependent type theory of an arithmetic universe.

Keywords: Dependent type theory, categorical logic, pretoposes, sketches, sketch models.

1 Introduction

In category theory the notion of sketch is the categorical counterpart of the notion of formal deductive system in terms of commented graph. There exist also the corresponding notions of "theory" and "model" in terms of categories and sketch morphisms (see for example [1,5]).

In this paper we consider finite decidable FP-sketches internal to an arithmetic universe. By a FP-sketch we mean a sketch with terminal and binary product cones. By an arithmetic universe we mean a list-arithmetic pretopos. Indeed, in [9] we proposed the notion of list-arithmetic pretopos as the general categorical definition for the construction of arithmetic universe performed by Andrè Joyal to provide a categorical proof of Gödel incompleteness

doi:10.1016/j.entcs.2004.06.054

theorems. This definition is justified in [9] by the fact that Joyal's universes are in particular list-arithmetic pretoposes and an initial list-arithmetic pretopos is equivalent to an initial Joyal's arithmetic universe and also by the fact that in any list-arithmetic pretopos we can build free internal categories and diagrams as in Joyal's universes (see [10]).

Here we carry on with testing the expressiveness of list-arithmetic pretoposes by proving a constructive version of Ehresmann-Kennison's theorem for finite decidable FP-sketches, as conjectured by S. Vickers and others working in the field.

Ehresmann-Kennison's theorem (see [1] page 146) states that the category of models of small sketches, with values in the category Set of ZFC sets, is reflective in the corresponding category of graph morphisms.

Here, we build a reflection of graph morphisms on finite decidable FP-sketches with values in an arithmetic universe into the corresponding category of models.

To define a finite decidable FP-sketch internal to an arithmetic universe and to perform the proof of the reflection we employ an internal language for list-arithmetic pretoposes formulated as a dependent type theory in the style of Martin-Löf's type theory (see [8]). This means that we can treat any arithmetic universe as a syntactic category built out of its internal language.

For the finite case, the proof we give turns out to be a constructive and predicative formulation of the classical proof regarding models of a small sketch in Set.

Indeed, the proof in [1] employs Freyd's adjoint functor theorem and quantification over all models, while our proof for finite decidable FP-sketches must use only predicative coherent logic - which does not have implication and universal quantification - together with lists and quotients, namely the only logical and set-theoretic constructors of the internal type theory of an arithmetic universe. The key point in our proof is to perform some inductive definitions to build trees on the arrows of the considered sketch and values of the graph-morphism which must be turned into a model.

The theorem proved here can be extended also to finite decidable lex-sketches by building more complicated trees and can be applied to build theories of such sketches as in [1] but considering an arithmetic universe as our set-theoretic universe in place of the category of Sets.

Our ultimate hope is that these results, possibly extended to finite decidable lex-sketches with also coproducts, could be useful for applications to database modelling as presented in [6]. The reason is that working within an intuitionistic predicative universe as our set-theoretic universe forces us to perform more effective constructions based on more elementary properties

than working within the category of Sets.

2 Sketches and models within an arithmetic universe

We recall the definition of arithmetic universe from [9].

Definition 2.1 An *arithmetic universe* [1] is a list-arithmetic pretopos, that is a pretopos (see also [14], [4]) with parameterized list objects (see also [2]).

In [8] we proved that the dependent typed calculus $\mathcal{A}_u$ formulated in the style of Martin-Löf's type theory (see the appendix) provides an internal language for list-arithmetic pretoposes, that is for arithmetic universes.

In few words this internal language for arithmetic universes corresponds to predicative coherent logic equipped with the set-theoretic constructions of lists and quotients. With respect to intuitionistic logic, predicative coherent logic lacks of implication and universal quantification, namely it has only conjunction, falsum, disjunction and existential quantification.

From [10] we recall that an arithmetic universe has also coequalizers:

Proposition 2.2 *In any list-arithmetic pretopos $\mathcal{U}$ there exists the coequalizer of any two given morphisms.*

From the type-theoretic or logical point of view this proposition says that in the dependent typed calculus $\mathcal{A}_u$ for arithmetic universes we can define the quotient type on any relation that is not necessarily an equivalence relation. We will make use of this property quite often in the constructions performed in this paper.

Given an arithmetic universe $\mathcal{U}$ we will introduce the notion of finite decidable FP-sketch internally to $\mathcal{U}$ by using the typed calculus $\mathcal{A}_u$ of the appendix.

A generic sketch is a graph with a set of identities, of diagrams, of cones and of cocones (for a general account on sketches and their models we refer to [1,5].)

The notion of sketch is the categorical counterpart of the notion of formal deductive system in terms of commented graph. By means of sketches we can specify algebraic structures like monoids, groups, rings in a categorical way.

Here, we restrict our attentions to finite sketches with terminal and binary product cones and no cocones, called finite FP-sketches. Actually we use the same terminology as for sketches with finite product cones, not necessarily only binaries, since we can always define a finite product cone through binary ones. Since by a finite set we mean a set in a bijective correspondence with the set

[1] Note that in [18] an apparently weaker general categorical definition of arithmetic universe is used.

of natural numbers minor to a given natural number, our notion of finiteness within an arithmetic universe implies the decidability of the equality of the elements of our finite set and this is why we add the adjective "decidable".

Before starting to give the needed definitions about sketches, we make the following remarks.

Remark 2.3 By means of the internal language $\mathcal{A}_u$, we identify an object A of an arithmetic universe $\mathcal{U}$ with a type A and a morphism $f : A \to B$ in $\mathcal{U}$ with a term $f(x) \in B\ [x \in A]$. Conversely, any closed type built out of the rules of $\mathcal{A}_u$ in the appendix denotes an object of $\mathcal{U}$ and any term of the form $f(x) \in B\ [x \in A]$, with A, B closed types, denotes a morphism of $\mathcal{U}$. For a more precise correspondence between categories and type theories see [8].

Remark 2.4 By means of the internal language we can prove that a morphism $f : A \to B$ of $\mathcal{U}$ is monic if and only if we can derive a proof of the type

$$x =_A y\ [x \in A, y \in A, z \in f(x) =_B f(y)]$$

where $x =_A y\ [x \in A, y \in A]$ stands for the equality type $\mathsf{Eq}(A, x, y)$ in the appendix.

Definition 2.5 We say that two types A, B in the internal language of $\mathcal{U}$ are isomorphic, indicated with $A \simeq B$, if there exist two terms $f(x) \in B\ [x \in A]$ and $g(y) \in A\ [y \in B]$ such that we can derive a proof of $g(f(x)) =_A x\ [x \in A]$ and a proof of $f(g(y)) =_A y\ [y \in A]$

Definition 2.6 A **finite decidable FP-sketch** $\mathcal{S} \equiv (\mathcal{G}, Diag, Con_{FP})$ internal to the arithmetic universe $\mathcal{U}$ is given with following data:

(i) a graph $\mathcal{G} \equiv (G_0, G_1, \delta_o, \delta_1)$ in $\mathcal{U}$, that is G_0 and G_1 are two objects of $\mathcal{U}$, denoting respectively the set of objects and of arrows, with δ_o and δ_1 morphisms of $\mathcal{U}$, denoting respectively the domain and codomain maps

$$G_1 \underset{\delta_1}{\overset{\delta_o}{\rightrightarrows}} G_0$$

such that there exists two natural numbers n_0, n_1 for which we can prove that the graph is finite (and hence decidable) in the sense that

$$G_0 \simeq \Sigma_{x \in N}\ x \leq n_0 \qquad G_1 \simeq \Sigma_{x \in N}\ x \leq n_1$$

(ii) a monic unit map sending any object into its unit arrow

$$\mathsf{id}_{(-)} : G_0 \rightarrowtail G_1$$

(iii) a set of diagrams given by a mono

$$Diag : D_0 \rightarrowtail \Sigma_{z \in Cat(\mathcal{G})_1 \times Cat(\mathcal{G})_1}\ \widehat{\delta_0}(\pi_1(z)) =_{G_0} \widehat{\delta_0}(\pi_2(z)) \quad \times \quad \widehat{\delta_1}(\pi_1(z)) =_{G_0} \widehat{\delta_1}(\pi_2(z))$$

where $Cat(\mathcal{G})$ is the free category generated from the graph $\mathcal{G}$ defined as in [10] with $\widehat{\delta_0}, \widehat{\delta_1} : Cat(\mathcal{G})_1 \to Cat(\mathcal{G})_0$ the domain and codomain maps;

(iv) a set of terminal and product cones described by $Con_{FP} \equiv (V, \nu, con_{FP})$ where V is the set of vertexes of the product and terminal cones and ν is a morphism

$$\nu : V \longrightarrow G_0$$

mapping the cone vertexes into objects of the graph and the cones on any $v \in V$ are described by

$$con_{FP}(v) \in (\Sigma_{f_1 \in G_1}\ \Sigma_{f_2 \in G_1}\quad \delta_0(f_1) =_{G_0} \nu(v)\ \times\ \delta_0(f_2) =_{G_0} \nu(v))\ \oplus\ \top$$

Remark 2.7 In the following, for simplicity, we suppose that $V \equiv \top \oplus P$ with

$$con_{FP}(\mathsf{inl}(*)) \equiv \mathsf{inr}(*) \qquad con_{FP}(\mathsf{inr}(p)) \equiv \mathsf{inl}(< \pi_{a(p)}, < \pi_{b(p)}, < \mathsf{eq}, \mathsf{eq} >>)$$

with $a(p) = \delta_1(\pi_{a(p)})$ and $b(p) = \delta_1(\pi_{b(p)})$. We will also use the following abbreviations: $v_t \equiv \mathsf{inl}(*)$ and $v_p \equiv \mathsf{inr}(p)$.

We can consider graph-morphisms (see [1,5]) from the internal sketch $\mathcal{S}$ to the underlying sketch of the arithmetic universe $\mathcal{U}$ itself by means of internal diagrams on the graph of $\mathcal{S}$:

Definition 2.8 A **graph morphism** on a finite decidable FP-sketch $\mathcal{S} \equiv (\mathcal{G}, Diag, Con_{FP})$ in $\mathcal{U}$, indicated with $F : \mathcal{S} \to \mathcal{U}$, is an internal diagram on the internal graph $\mathcal{G}$ in $\mathcal{U}$ (for a categorical definition see for example [10]), that is in the internal language of $\mathcal{U}$ we can derive a dependent type $F_0(x)$ $[x \in G_0]$ and a typed term

$$F_1(f)(z) \in F_0(y)$$
$$[f \in G_1,\ x \in G_0,\ z \in F_0(x),\ y \in G_0,\ z' \in x =_{G_0} \delta_0(f),\ z'' \in y =_{G_0} \delta_1(f)]$$

describing a graph-morphism from the underlying graph of the sketch to that of the arithmetic universe sending objects to objects, arrows to arrows in a way as to preserve their domain and codomain.

Definition 2.9 Given a finite decidable FP-sketch $\mathcal{S} \equiv (\mathcal{G}, Diag, Con_{FP})$ in $\mathcal{U}$, a **natural transformation** $\alpha : F \to H$ from the graph morphism $F : \mathcal{S} \to \mathcal{U}$ to the graph morphism $H : \mathcal{S} \to \mathcal{U}$ in $\mathcal{U}$ is given by a term

$$\alpha_x(z) \in H_0(x)\ [x \in G_0, z \in F_0(x)]$$

such that we can prove that

$$H_1(f)(\alpha_x(z)) =_{H_0(y)} \alpha_y(F_1(f)(z))$$
$$[x \in G_0,\ y \in G_0,\ f \in G_1,\ z \in F_0(x),\ z' \in x =_{G_0} \delta_0(f),\ z'' \in y =_{G_0} \delta_1(f)]$$

Now, we introduce the concept of a model for a FP-sketch in the arithmetic universe $\mathcal{U}$.

Definition 2.10 A **model** for a FP-sketch $\mathcal{S} \equiv (\mathcal{G}, Diag, Con_{FP})$ in the arithmetic universe $\mathcal{U}$ is a graph-morphism $M : \mathcal{S} \to \mathcal{U}$ in $\mathcal{U}$ represented by $M_0(x)$ $[x \in G_0]$ and a typed term $M_1(f)(z) \in M_0(y)$ $[f \in G_1, x \in G_0, z \in M_0(x), y \in G_0, z' \in x =_{G_0} \delta_0(f), z'' \in y =_{G_0} \delta_1(f)]$ such that

(i) unit maps of $\mathcal{S}$ are sent to the corresponding unit maps of $\mathcal{U}$, that is we can derive a proof of

$$M_1(\mathsf{id}_x)(z) =_{M_0(x)} z \ [x \in G_0, z \in M_0(x)]$$

(ii) diagrams of $\mathcal{S}$ are sent to commutative diagrams in $\mathcal{U}$, that is we can derive a proof of

$$\widetilde{M}_1(\pi_1(Diag(d))) =_{M_0(\widehat{\delta_1}(\pi_1(Diag(d))))} \widetilde{M}_1(\pi_2(Diag(d))) \ [d \in D_0]$$

where $\widetilde{M}$ is the free internal categorical diagram [2] generated from M on the free internal category $Cat(\mathcal{S})$ generated from the sketch graph $\mathcal{G}$ (for the construction of the free internal categorical diagram $\widetilde{M}$ see [10]);

(iii) cones of $\mathcal{S}$ are sent to cone limits in $\mathcal{U}$, that is , if $V \equiv \top \oplus P$ as in remark 2.7, then $M_0(\nu(v_t)))$ is isomorphic to the terminal object $\top$ and for every $p \in P$ then $M_0(\nu(v_p))$ is isomorphic to $M_0(a(p)) \times M_0(b(p))$ and if $f(z) \in M_0(\nu(v_p))$ $[z \in M_0(a(p)) \times M_0(b(p))]$ is the isomorphism then we derive a proof of

$$M_1(\pi_{a(p)})(f(z)) =_{M_0(a(p))} \pi_1(z) \ [p \in P, z \in M_0(a(p)) \times M_0(b(p))]$$

and a proof of

$$M_1(\pi_{b(p)})(f(z)) =_{M_0(b(p))} \pi_2(z) \ [p \in P, z \in M_0(a(p)) \times M_0(b(p))]$$

that is projections are sent to projections.

Now we give the definitions of the category of graph-morphisms and of models into an arithmetic universe. For what follows we assume that $\mathcal{S} \equiv (\mathcal{G}, Diag, Con_{FP})$ is a finite decidable FP-sketch in $\mathcal{U}$.

Definition 2.11 $\mathcal{U}^{\mathcal{S}}$ is the category of graph morphisms $F : \mathcal{S} \to \mathcal{U}$ as objects and natural transformations as morphisms with the obvious composition and units.

Then we define the category of graph morphisms preserving the diagrams:

[2] For the definition of internal categorical diagrams on an internal category see, for example, page 242 of [12], where they are simply called internal diagrams. Here, we add the adjective "categorical" to distinguish internal categorical diagrams on a category preserving units and its internal composition from internal diagrams on the underlying graph.

Definition 2.12 $D(\mathcal{S},\ \mathcal{U})$ is the full subcategory of the category $\mathcal{U}^{\mathcal{S}}$ with graph morphisms sending diagrams of the sketch into commutative diagrams of $\mathcal{U}$ and units to units as in point (i) and (ii) of definition 2.10.

Definition 2.13 $Mod(\mathcal{S},\ \mathcal{U})$ is the full subcategory of the category $\mathcal{U}^{\mathcal{S}}$ with models for the decidable finite FP-sketch $\mathcal{S}$ in the arithmetic universe $\mathcal{U}$.

At this point note that working with graph morphisms on an internal graph is not much different than working with the corresponding internal categorical diagrams on the free category generated from the graph sending units to units and preserving the internal composition. Indeed, thanks to the fact (shown in [10]) that any graph morphism on a internal graph $\mathcal{G}$ gives rise to a free internal categorical diagram on the category $Cat(\mathcal{G})$, we can easily prove that:

Proposition 2.14 *The category of graph morphisms $\mathcal{U}^{\mathcal{S}}$ is equivalent to the category of internal categorical diagrams on $Cat(\mathcal{S})$ in $\mathcal{U}$.*

3 The reflection

Here, we will introduce all the constructions necessary to prove our main theorem, namely that given a finite decidable FP-sketch $\mathcal{S}$, there exists a reflection of $\mathcal{U}^{\mathcal{S}}$ into $Mod(\mathcal{S},\ \mathcal{U})$.

This result will be reached in three steps:

(i) first we observe that $\mathcal{D}(\mathcal{S},\mathcal{U})$ is reflective into $\mathcal{U}^{\mathcal{S}}$;

(ii) then we prove that $Mod(\mathcal{S},\ \mathcal{U})$ is reflective into $\mathcal{D}(\mathcal{S},\mathcal{U})$, which is the hard part [3];

(iii) finally, we compose the two reflections to obtained the claimed reflection of $\mathcal{U}^{\mathcal{S}}$ into $Mod(\mathcal{S},\ \mathcal{U})$.

Here is the first step.

Proposition 3.1 *Given a FP-sketch $\mathcal{S}$, the inclusion functor*

$$\mathcal{I} : \mathcal{D}(\mathcal{S},\mathcal{U}) \to \mathcal{U}^{\mathcal{S}}$$

has got a left adjoint.

Proof. The construction of the left adjoint is done by mimicking the usual set-theoretic construction. To each graph morphism F we associate the graph

[3] This reflection corresponds to Ehresmann-Kennison's theorem in [1] for sketches in Set, since in [1] graph morphisms are supposed to preserve also the specific diagrams of the considered sketch and their units. Here, we prefer to call graph morphisms those diagrams that just preserve the graph structure as in def. 2.8.

morphism $\tilde{F}$ defined on each object $x \in G_0$ by coequalizing the morphisms obtained by applying F to all the pairs of composable sequences of morphisms in the sketch diagrams or in the unit diagrams possibly extended by composing them with a sequence of morphisms with codomain x. Such an object $\tilde{F}(x)$ can be built thanks to prop 2.2. □

In order to perform the second step, namely to prove the reflection of $\mathcal{D}(\mathcal{S},\mathcal{U})$ into $Mod(\mathcal{S},\ \mathcal{U})$ we need to build some inductive types inside $\mathcal{U}$, which are partly based on sequences of composable arrows of $\mathcal{S}$ quotiented under the diagrams.

Therefore, we will work with the smallest category making the diagrams of $\mathcal{S}$ commute. This is obtained by quotienting the morphisms $Cat(\mathcal{S})_1$ of the free category $Cat(\mathcal{S})$ generated by the sketch graph $\mathcal{G}$ (described in [10]) under the diagrams of $\mathcal{S}$, which is possible thanks to prop 2.2.

Proposition 3.2 *In $\mathcal{U}$ we can define the quotient category*

$$\mathcal{D}_1 \equiv (Cat(\mathcal{S})_1/Diag^*) \quad \text{and} \quad \mathcal{D}_0 \equiv G_0$$

where $Diag^*$ is defined as the union of the sketch diagrams indexed by $Diag$ with the diagrams identifying the identities in $Cat(\mathcal{S})$ with the sketch identities. We denote with $[-] : \mathcal{S} \to Cat(\mathcal{S})/Diag^*$ the graph morphism projecting the sketch into the category.

Warning on notation. Note that in the following, in order to make the formulas more readable, we will use the abbreviations: for $a, b \in G_0$

$$\mathcal{D}_1(a,b) \equiv \Sigma_{x \in \mathcal{D}_1}\ \delta_0(x) =_{G_0} a\ \times\ \delta_1(x) =_{G_0} b$$

and given $f \in G_1$ we will simply write

$$f \in \mathcal{D}_1(a,b) \quad \text{instead of} \quad < [f], < \mathsf{eq}, \mathsf{eq} >> \in \mathcal{D}_1(a,b)$$

3.1 The construction of trees

In this section we describe the construction of trees that we will use to build a model for a finite decidable FP-sketch starting from a graph morphism on the considered sketch. The trees are made up of the following ingredients:

- composable sequences of morphisms of the given finite decidable FP-sketch $\mathcal{S}$ internal to $\mathcal{U}$;
- values of a graph morphism $F : \mathcal{S} \to \mathcal{U}$ in $\mathcal{D}(\mathcal{S},\mathcal{U})$ applied to a sketch object.

In the following we will often use the notation $F(x)$ on an object $x \in G_0$ or $F(f)$ on an arrow $f \in G_1$ instead of writing $F_0(x)$ or $F_1(f)$.

3.1.1 The idea

The basic idea is to be able to build a model $\widehat{F}$ out of a graph morphism $F : \mathcal{S} \to \mathcal{U}$. Let us start by considering to have a sketch with only a terminal cone with vertex $\nu(v_t) \in G_0$. Therefore we expect to define a unit $\eta : F \to \widehat{F}$. Since we know that $\widehat{F}(\nu(v_t))$ must be a terminal object, for simplicity we suppose $\widehat{F}(\nu(v_t)) \equiv \top$. Then, the unit naturality diagram for an arrow ending in the terminal vertex trivially commutes. Instead, if we consider an arrow like $g : \nu(v_t) \to a$ in G_1, then from the commutativity of the following diagram

$$\begin{array}{ccc} F(\nu(v_t)) & \xrightarrow[\eta_{\nu(v_t)}]{} & \top \\ {\scriptstyle F(g)}\downarrow & & \downarrow{\scriptstyle \widehat{F}(g)} \\ F(a) & \xrightarrow[\eta_a]{} & \widehat{F}(a) \end{array}$$

we get that $\widehat{F}(a)$ must be something like the push-out of the diagram, that is

$$\widehat{F}(a) \equiv (F(a) \oplus \mathcal{D}_1(\nu(v_t), a))/Rel$$

where Rel must include that $F(g)(y)$ for $y \in F(\nu(v_t))$ is in relation with g. In this case we put $\widehat{F}(g)(*) \equiv [\mathsf{inr}(g)]$ and $\eta_a(x) \equiv [\mathsf{inl}(x)]$ for $x \in F(a)$ (where inr and inl are the injections in the sum type in the appendix).

Then, consider a sketch with a product cone of vertex $c \times d$ and an arrow $g : c \times d \to a$ and suppose for simplicity that $\widehat{F}(c \times d) = \widehat{F}(c) \times \widehat{F}(d)$ and that $\eta_{\nu(c\times d)} \equiv \eta_c \times \eta_d \cdot < F(\pi_c), F(\pi_d) >$. Then, from the commutativity of the diagram

$$\begin{array}{ccc} F(c\times d) & \xrightarrow[\eta_{\nu(c\times d)}]{} & \widehat{F}(c)\times\widehat{F}(d) \\ {\scriptstyle F(g)}\downarrow & & \downarrow{\scriptstyle \widehat{F}(g)} \\ F(a) & \xrightarrow[\eta_a]{} & \widehat{F}(a) \end{array}$$

recalling that $\widehat{F}(c)$ must include $F(c) \oplus \mathcal{D}_1(\nu(v_t), c))/Rel$ and $\widehat{F}(d)$ must include $F(d) \oplus \mathcal{D}_1(\nu(v_t), d))/Rel$, we conclude that $\widehat{F}(a)$ must include all the combinations of the product of each member of $\widehat{F}(c)$ with each member of $\widehat{F}(d)$. Hence, $\widehat{F}(a)$ must include not only $F(a)$ but, for example, also elements $< (x, y), g >$ with $x \in F(c)$ and $y \in F(d)$. In this case, in order to make the diagram commute, the element $< (F(\pi_{a(p)})(z), F(\pi_{b(p)})(z)), g >$ has to be put in relation with any $F(g)(z)$ for $z \in F(c \times d)$. Moreover, $\widehat{F}(a)$ must include $< (x, f_2), g >$ with $x \in F(a(p))$ and $f_2 : \nu(v_t) \to d$, or $< (f_1, y), g >$ with $y \in F(b(p))$ and $f_1 : \nu(v_t) \to c$, or $< (f_1, f_2), g >$ with $f_1 : \nu(v_t) \to c$ and $f_2 : \nu(v_t) \to d$. These observations inspired us to define $\widehat{F}(a)$ by means of trees made up of sketch arrows and values of the graph morphism F on a sketch object. The trees are inductively generated as follows. An element $*$

of the terminal object $\top$ is a base tree of vertex $v \equiv v_t$

$$(*)$$

and also a couple

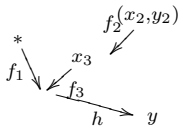
$$(x, y)$$

with $x \in F(a(p))$ and $y \in F(b(p))$ is a base tree with vertex $v \equiv v_p$. Then, we may extend a tree w with vertex v with an arrow $f : v \to a(p)$ to form a left branch of a new tree, whose right branch may be simply some $y \in F(b(p))$, like

or we may extend the tree w with vertex v with an arrow $f : v \to a(p)$ to form a left branch of a new tree, whose right branch is another tree w' with vertex v' extended with an arrow $g : v' \to b(p)$, like

or we may extend the given tree w with vertex v with an arrow $g : v \to b(p)$ to form a right branch of a new tree, whose left branch may be simply some $x \in F(b(p))$, like

One of the resulting trees may be



In the case $F \equiv M$ is already a model, then this tree with vertex p corresponds to the point of $M(p)$

$$< M(h)(z_1), y >$$

where $z_1 =< M(f_1)(*), M(f_3)(z_2) >$ - with $*$ the unique element in the terminal object $M(\nu(v_t))$ - and $z_2 =< x_3, M(f_2)(< x_2, y_2 >) >$. In practice with such trees we keep trace of points and arrows which could be applied properly to get a point if the given graph morphism F were a model.

3.1.2 Trees and related relations

Here, we show how to build the needed trees internally to an arithmetic universe. Although we do not have types of well-founded trees [15] in the calculus

$\mathcal{A}_u$ for arithmetic universes, by making an essential use of lists we can define some useful inductive types, as shown in the following proposition, to meet our purpose. Recall that N *type* is the type of natural numbers.

Proposition 3.3 *In the internal type theory of the arithmetic universe* $\mathcal{U}$ *we can define a type*

$$\Sigma_{n\in N}\, R(n,s) \text{ type } [s \in S]$$

provided that $R(0,s)$ *type* $[s \in S]$ *is derivable in the calculus and that*

$$R(n+1,s) \equiv \Sigma_{s'\in S}\, R(n,s') \;\&\; H(s',s)$$

for $s \in S, n \in N$ *where* $H(s',s)$ *type* $[s \in, s' \in S]$ *is derivable in the calculus.*

Moreover, we can argue by induction on the type according to the following elimination and conversion rules:

$$E)\quad \frac{\begin{array}{ll} C(s,z) \text{ type} & [s\in S, z\in \Sigma_{n\in N}\, R(n,s)] \\ c_o(s,d)\in C(s,<0,d>) & [s\in S, d\in R(0,s)] \\ c_1(s,s',z',h,u)\in C(s,<n+1,<s',<z',h>>>) & \\ \qquad [s\in S, s'\in S, n\in N, z'\in R(n,s'), h\in H(s',s), u\in C(s',<n,z'>)] & \end{array}}{\mathbf{El}(c_0,c_1,s,z)\in C(s,z) \qquad [s\in S, z\in \Sigma_{n\in N}\, R(n,s)]}$$

$$\mathbf{El}(c_0,c_1,s,<0,d>) = c_0(s,d) \in C(s,<0,d>)$$

$$\mathbf{El}(c_0,c_1,s,<n+1,<s',<z',h>>>) = c_1(s,s',z',h,\mathbf{El}(c_0,c_1,s',<n,z'>)) \in C(s,<n+1,<s',<z',h>>>)$$

Proof. The candidate type is

$$R(0,s) \;\oplus\; (\,\Sigma_{w\in List^*(\Sigma_{x\in S}\ \Sigma_{y\in S} H(x,y))}\quad R(0,\,\widetilde{\pi_1}(p_1(w)))\,) \;\&\; \widetilde{\pi_2}(p_{lh(w)}(w)) =_S s$$
$$\&\quad \overline{\pi_1}(front(w)) =_{List(\Sigma_{x\in S}\ \Sigma_{y\in S} H(x,y))} \overline{\pi_2}(back(w))\,)$$

where in general $List^*(A)$ denotes the type of non-empty lists of A, $p_n(w)$ is the projection of the n-th element of the list w, $lh(w)$ is the length of the list w, $\overline{\pi_1}$ is the lifting on lists of the first projection $\widetilde{\pi_1}(z) \equiv \pi_1(z) \in S$ for $z \in \Sigma_{x\in S}\ \Sigma_{y\in S} H(x,y)$, $\overline{\pi_2}$ is the lifting on lists of the second projection $\widetilde{\pi_2}(z) \equiv \pi_1(\pi_2(z)) \in S$ for $z \in \Sigma_{x\in S}\ \Sigma_{y\in S} H(x,y)$ and $front(w)$ takes out the first element from the list w by leaving the front of the list, while $back(w)$ takes out the last element from the list w by leaving the remaining tail. □

Thanks to prop. 3.3 we can prove:

Lemma 3.4 (Trees) *In the internal type theory of* $\mathcal{U}$ *we can define a type*

$$Tree(F,v)\ [v\in V]$$

with the following introduction rules to form its terms:

$$(*) \in Tree(F, v_t)$$

$$\frac{p \in P \qquad x \in F_0(a(p)) \qquad y \in F_0(b(p))}{(x, y) \in Tree(F, v_p)}$$

$$\frac{v \in V \quad p \in P \quad x \in F_0(a(p)) \quad w \in Tree(F, v) \quad f \in \mathcal{D}_1(\nu(v), b(p))}{(x, w/f) \in Tree(F, v_p)}$$

$$\frac{v \in V \quad p \in P \quad w \in Tree(F, v) \quad f \in \mathcal{D}_1(\nu(v), a(p)) \quad y \in F_0(b(p))}{(w/f, y) \in Tree(F, v_p)}$$

$$\frac{\begin{array}{c} v_1 \in V \quad v_2 \in V \quad p \in P \quad w_1 \in Tree(F, v_1) \quad w_2 \in Tree(F, v_2) \\ f_1 \in \mathcal{D}_1(\nu(v_1), a(p)) \qquad f_2 \in \mathcal{D}_1(\nu(v_2), b(p)) \end{array}}{(w_1/f_1, w_2/f_2) \in Tree(F, v_p)}$$

Moreover, on this type we can argue by induction thanks to the corresponding elimination and conversion rules formulated according to the style of Martin-Löf's type theory [15] as in prop. 3.3.

Then, always thanks to prop. 3.3, we can define a type containing branches of the defined trees possibly extended with arrows ending on any object of the sketch or simply values of F on a sketch object.

Lemma 3.5 (Extended branches) *In the internal type theory of $\mathcal{U}$ we can define a type*

$$Bran(F, x) \ [x \in G_0]$$

with the following introduction rules to form its terms:

$$\frac{x \in G_0 \quad u \in F_0(x)}{/u \in Bran(F, x)}$$

$$\frac{x \in G_0 \quad v \in V \quad w \in Tree(F, v) \quad f \in \mathcal{D}_1(\nu(v), x)}{w/f \in Bran(F, x)}$$

Moreover, on this type we can argue by induction thanks to the corresponding elimination and conversion rules formulated in the style of Martin-Löf's type theory [15] as in prop. 3.3.

In addition we can define an operation composing a left branch with an appropriate right branch to form a tree:

Lemma 3.6 *In the internal type theory of $\mathcal{U}$ we can define an operation*

$$(z_1, z_2) \in Tree(F, v_p) \ [p \in P, z_1 \in Bran(F, a(p)), z_2 \in Bran(F, b(p))]$$

such that for any $p \in P$ and any tree $w \in Tree(F, v_p)$ we can derive a proof of

$$\Sigma_{z_1 \in Bran(F, a(p))} \Sigma_{z_2 \in Bran(F, b(p))} \, (z_1, z_2) =_{Tree(F, v_p)} w$$

Then, always thanks to prop. 3.3, we define a relation on branches which will help to properly define the reflector:

Lemma 3.7 (Branches relation) *In the internal type theory of $\mathcal{U}$ we can define the inductive type*

$$z_1 \simeq_{Bran(F)} z_2 \ [x \in G_0, z_1 \in Bran(F,x), z_2 \in Bran(F,x)]$$

whose formation rules are the following:

$$\mathsf{Ter}_1 \quad \frac{y \in F_0(\nu(v_t))}{(*)/\mathsf{id}_{\nu(v_t)} \simeq_{Bran(F)} y}$$

$$\mathsf{Ter}_2 \quad \frac{v \in V \qquad w \in Tree(F,v) \quad f_1 \in \mathcal{D}_1(\nu(v),\nu(v_t))}{w/f_1 \simeq_{Bran(F)} (*)/\mathsf{id}_{\nu(v_t)}}$$

$$\mathsf{Prod}_1 \quad \frac{p \in P \qquad bl \in Bran(F,a(p)) \qquad br \in Bran(F,b(p))}{(bl,br)/\pi_{a(p)} \simeq_{Bran(F)} bl}$$

$$\mathsf{Prod}_2 \quad \frac{p \in P \qquad bl \in Bran(F,a(p)) \qquad br \in Bran(F,b(p))}{(bl,br)/\pi_{b(p)} \simeq_{Bran(F)} br}$$

$$\mathsf{Prod}_3 \quad \frac{p \in P \qquad u \in F_0(\nu(v_p))}{(F_1(\pi_{a(p)})(u), F_1(\pi_{b(p)})(u))/\mathsf{id}_{\nu(v_p)} \simeq_{Bran(F)} u}$$

$$\mathsf{Prod}_4 \quad \frac{v \in V \qquad w \in Tree(F,v) \qquad p \in P \qquad g \in \mathcal{D}_1(\nu(v),\nu(v_p))}{(\,w/(\pi_{a(p)} \cdot g)\ ,\ w/(\pi_{b(p)} \cdot g)\,)/\mathsf{id}_{\nu(v_p)} \simeq_{Bran(F)} w/g}$$

$$\mathsf{Prod}_5 \quad \frac{\begin{array}{llll} p \in P & bl \in Bran(F,a(p)) & bl' \in Bran(F,a(p)) & br \in Bran(F,b(p)) \\ x \in G_0 & f \in \mathcal{D}_1(\nu(v_p),x) & bl \simeq_{Bran(F)} bl' & \end{array}}{(bl,br)/f \simeq_{Bran(F)} (bl',br)/f}$$

$$\mathsf{Prod}_6 \quad \frac{\begin{array}{llll} p \in P & bl \in Bran(F,a(p)) & br \in Bran(F,b(p)) & br' \in Bran(F,b(p)) \\ x \in G_0 & f \in \mathcal{D}_1(\nu(v_p),x) & br \simeq_{Bran(F)} br' & \end{array}}{(bl,br)/f \simeq_{Bran(F)} (bl,br')/f}$$

$$\mathsf{Comp}_1 \quad \frac{\begin{array}{lllll} v \in V & w \in Tree(F,v) & x \in G_0 & x' \in G_0 & u \in F_0(x') \\ f \in \mathcal{D}_1(\nu(v),x') & h \in \mathcal{D}_1(x',x) & w/f \simeq_{Bran(F)} u & & \end{array}}{w/h \cdot f \simeq_{Bran(F)} F_1(h)(u)}$$

$$\mathsf{Comp}_2 \quad \frac{\begin{array}{lllll} v_1 \in V & w_1 \in Tree(F,v_1) & v_2 \in V & w_2 \in Tree(F,v_2) & x' \in G_0 \qquad x \in G_0 \\ f_1 \in \mathcal{D}_1(\nu(v_1),x') & f_2 \in \mathcal{D}_1(\nu(v_2),x') & h \in \mathcal{D}_1(x',x) & w_1/f_1 \simeq_{Bran(F)} w_2/f_2 & \end{array}}{w_1/h \cdot f_1 \simeq_{Bran(F)} w_2/h \cdot f_2}$$

Moreover, on this type we can argue by induction thanks to the corresponding elimination and conversion rules formulated in the style of prop. 3.3.

Now, to define the object part of the reflector, we use the type $T(F)(x)$ of extended trees on any $x \in G_0$ defined as the coproduct of trees extended with a sketch arrow with a copy of $F_0(x)$:

$$T(F)(x) \equiv F_0(x) \ \oplus \ \Sigma_{v \in V} \ Tree(F,v) \ \times \ \mathcal{D}_1(\nu(v), x)$$

Note that in the following we will use this abbreviation: for $x \in G_0$, $v \in V$, $w \in Tree(F,v)$, $f \in \mathcal{D}_1(\nu(v), x)$

$$< v, w, f > \equiv < v, < w, f >> \in \Sigma_{v \in V} \ Tree(F,v) \ \times \ \mathcal{D}_1(\nu(v), x)$$

Then, on the extended trees we define a relation based on the branches relation:

Definition 3.8 [Extended trees relation] In the internal language of $\mathcal{U}$ we define the type

$$z \simeq_{T(F)} z' \ [x \in G_0, z \in T(F)(x), z' \in T(F)(x)]$$

as follows: for $x \in G_0, z \in T(F)(x), z' \in T(F)(x)$

$$\begin{aligned} z \simeq_{T(F)} z' \equiv \quad & (\ \Sigma_{u \in F_0(x)} \ z =_{T(F)(x)} [\mathsf{inl}(u)] \\ & \& \quad \Sigma_{v \in V} \ \Sigma_{w \in Tree(F,v)} \ \Sigma_{f \in \mathcal{D}_1(\nu(v),x)} \quad z' =_{T(F)(x)} [\mathsf{inr}(< v, w, f >)] \\ & \& \quad w/f \simeq_{Bran(F)} u \ ) \\ & \oplus \\ & (\ \Sigma_{v_1 \in V} \ \Sigma_{v_2 \in V} \ \Sigma_{w_1 \in Tree(F,v_1)} \ \Sigma_{w_2 \in Tree(F,v_2)} \ \ \Sigma_{f_1 \in \mathcal{D}_1(\nu(v_1),x)} \ \Sigma_{f_2 \in \mathcal{D}_1(\nu(v_2),x)} \\ & z =_{T(F)(x)} [\mathsf{inr}(< v_1, w_1, f_1 >)] \quad \& \quad z' =_{T(F)(x)} [\mathsf{inr}(< v_2, w_2, f_2 >)] \\ & \& \qquad w_1/f_1 \simeq_{Bran(F)} w_2/f_2 \ ) \end{aligned}$$

3.2 The reflector

The definition of the reflector functor $\widehat{(-)} : D(\mathcal{S},\mathcal{U}) \to Mod(\mathcal{S}, \ \mathcal{U})$ is the following. For any graph morphism $F : \mathcal{S} \to \mathcal{U}$ in $\mathcal{D}(\mathcal{S},\mathcal{U})$ we put:

Definition 3.9 [$\widehat{F}$ on objects] For every $x \in G_0$ we define

$$\widehat{F}_0(x) \equiv \frac{F_0(x) \oplus \Sigma_{v \in V} \ Tree(F,v) \ \times \ \mathcal{D}_1(\nu(v), x)}{\simeq_{T(F)}}$$

Definition 3.10 [$\widehat{F}$ on morphisms] For any $g \in G_1$ such that $\delta_0(g) = x \in G_0$ and $\delta_1(g) = y \in G_0$ we define

$$\widehat{F}_1(g) : \widehat{F}_0(x) \longrightarrow \widehat{F}_0(y)$$

by the elimination rule on the quotient type $\widehat{F}_0(x)$: for every $z \in T(F)(x)$

$$\widehat{F}_1(g)([z]) \equiv \begin{cases} [\mathsf{inl}(F_1(g)(y))] & \text{if } z = \mathsf{inl}(y) \text{ for } y \in F_0(x) \\ [\mathsf{inr}(< v, w, g \cdot f >)] & \text{if } z = \mathsf{inr}(< v, w, f >) \text{ for } v \in V \\ & \quad w \in Tree(F, v) \text{ and } f \in \mathcal{D}_1(\nu(v), x) \end{cases}$$

Definition 3.11 [$\widehat{(-)}$ on natural transformations] Given a natural transformation $\alpha : F \longrightarrow H$ in $D(\mathcal{S}, \mathcal{U})$ we define a natural transformation

$$\widehat{\alpha} : \widehat{F} \longrightarrow \widehat{H}$$

in $Mod(\mathcal{S}, \mathcal{U})$ by making use of the following term

$$\widetilde{\alpha}(w) \in Tree(H, v) \; [v \in V, w \in Tree(F, v)]$$

defined by induction on the tree w as follows:

$$\widetilde{\alpha}(w) \equiv \begin{cases} (*) & \text{if } w = (*) \\ (\alpha_{a(p)}(x), \alpha_{b(p)}(y)) & \text{if } w = (x, y) \text{ for } x \in F_0(a(p)),\ y \in F_0(b(p)) \text{ and } p \in P \\ (\alpha_{a(p)}(x), \widetilde{\alpha}(w')/h) & \text{if } w = (x, w'/h) \text{ for } x \in F_0(a(p)) \text{ and } v \in V \text{ and } p \in P \\ & \quad w' \in Tree(F, v) \text{ and } h \in \mathcal{D}_1(\nu(v), b(p)) \\ (\widetilde{\alpha}(w')/h, \alpha_{b(p)}(y)) & \text{if } w = (w'/h, y) \text{ for } y \in F_0(b(p)) \text{ and } v \in V \text{ and } p \in P \\ & \quad w' \in Tree(F, v) \text{ and } h \in \mathcal{D}_1(\nu(v), a(p)) \\ (\widetilde{\alpha}(w_1)/h_1, \widetilde{\alpha}(w_1)/h_2) & \text{if } w = (w_1/h_1, w_2/h_2) \text{ and } v_1 \in V,\ v_2 \in V \text{ and } p \in P \\ & \quad w_1 \in Tree(F, v_1) \text{ and } h_1 \in \mathcal{D}_1(\nu(v_1), a(p)) \\ & \quad w_2 \in Tree(F, v_2) \text{ and } h_2 \in \mathcal{D}_1(\nu(v_2), b(p)) \end{cases}$$

Then we are ready to define $\widehat{\alpha}$ whose component on $x \in G_0$ is

$$(\widehat{\alpha})_x : \widehat{F}_0(x) \longrightarrow \widehat{H}_0(x)$$

defined by the elimination rule on the quotient type $\widehat{F}(x)$: for every $z \in T(F)(x)$

$$(\widehat{\alpha})_x([z]) \equiv \begin{cases} [\mathsf{inl}(\alpha_x(y))] & \text{if } z = \mathsf{inl}(y) \text{ for } y \in F_0(x) \\ [\mathsf{inr}(< v, < \widetilde{\alpha}(w), f >)] & \text{if } z = \mathsf{inr}(< v, w, f >) \text{ for } v \in V \\ & \quad w \in Tree(F, v) \text{ and } f \in \mathcal{D}_1(\nu(v), x) \end{cases}$$

Lemma 3.12 *$\widehat{F}$ is a model for the finite decidable FP-sketch $\mathcal{S}$ in $\mathcal{U}$.*

Proof. To show our statement is crucial to use the relation $\simeq_{Bran(F)}$ defined in lemma 3.7 on which the relation $\simeq_{T(F)}$ used in the definition of the object part of $\widehat{F}$ is based.

Just note that $\widehat{F}$ is a well defined graph morphism thanks to $\mathsf{Comp_1}$, $\mathsf{Comp_2}$ of $\simeq_{Bran(F)}$ in lemma 3.7, that $\widehat{F}_0(\nu(v_p))$ is a product for $p \in P$ and that $\widehat{F}_1(\pi_{a(p)})$ and $\widehat{F}_1(\pi_{b(p)})$ are projections thanks to $\mathsf{Prod_i}$ for $i = 1, .., 6$ and $\mathsf{Comp_i}$ for $i = 1, 2$ of $\simeq_{Bran(F)}$ in lemma 3.7, and that $\widehat{F}_0(\nu(v_t))$ is isomorphic to $\top$ thanks to $\mathsf{Ter_1}$, $\mathsf{Ter_2}$ of $\simeq_{Bran(F)}$ in lemma 3.7. □

Finally, we define the candidate to be the unit of the reflection as follows:

Definition 3.13 [Unit] Given a graph morphism $F : \mathcal{S} \to \mathcal{U}$ in $\mathcal{D}(\mathcal{S},\mathcal{U})$, we define a natural transformation

$$\eta_F : F \longrightarrow \widehat{F}$$

whose component $(\eta_F)_x : F_0(x) \longrightarrow \widehat{F}_0(x)$ on $x \in G_0$ is defined as follows: for every $z \in F_0(x)$

$$(\eta_F)_x(z) \equiv [\mathsf{inl}(z)]$$

Then we define the candidate to be the counit of the adjunction.

Definition 3.14 [Counit] Given a model $M \in Mod(\mathcal{S},\mathcal{U})$ we define the counit as a natural transformation

$$\varepsilon_M : \widehat{M} \longrightarrow M$$

by making use of the term

$$Ap(M)(w) \in M_0(\nu(v)) \ [v \in V, w \in Tree(M, v)]$$

defined by induction on the tree w as follows:

$$Ap(M)(w) \equiv \begin{cases} *_M & \text{if } w = (*) \\ < x, y > & \text{if } w = (x, y) \text{ with } x \in M_0(a(p)) \text{ and } y \in M_0(b(p)) \text{ for } p \in P \\ < x, M_1(h)(Ap(M)(w')) > & \text{if } w = (x, w'/h) \text{ with } x \in M_0(a(p)) \text{ and } w' \in Tree(M, v) \text{ and } h \in \mathcal{D}_1(\nu(v), b(p)) \text{ for } v \in V,\ p \in P \\ < M_1(h)(Ap(M)(w')), y > & \text{if } w = (w'/h, y) \text{ with } y \in M_0(b(p)) \text{ and } w' \in Tree(M, v) \text{ and } h \in \mathcal{D}_1(\nu(v), a(p)) \text{ for } v \in V,\ p \in P \\ < M_1(h_1)(Ap(M)(w_1)), M_1(h_2)(Ap(M)(w_2)) > & \text{if } w = (w_1/h_1, w_2/h_2) \text{ with } w_1 \in Tree(M, v_1) \text{ and } h_1 \in \mathcal{D}_1(\nu(v_1), a(p)) \text{ and } w_2 \in Tree(M, v_2) \text{ and } h_2 \in \mathcal{D}_1(\nu(v_2), b(p)) \text{ for } v_1 \in V,\ v_2 \in V,\ p \in P \end{cases}$$

where $*_M$ is the unique element in $M_0(\nu(v_t))$, which must be a terminal object in $\mathcal{U}$.
For every $x \in G_0$ we define

$$(\varepsilon_M)_x : \widehat{M_0}(x) \longrightarrow M_0(x)$$

by the elimination rule on the quotient type $\widehat{M_0}(x)$: for every $z \in T(M)(x)$

$$(\varepsilon_M)_x([z]) \equiv \begin{cases} y & \text{if } z = \mathsf{inl}(y) \text{ for } y \in M_0(x) \\ M_1(f)(Ap(M)(w)) & \text{if } z = \mathsf{inr}(< v, w, f >) \text{ for } v \in V,\ w \in Tree(M, v) \text{ and } f \in \mathcal{D}_1(\nu(v), x) \end{cases}$$

With all these definitions we conclude

Theorem 3.15 *Given a finite decidable FP-sketch $\mathcal{S}$, the inclusion functor $\mathcal{J} : Mod(\mathcal{S},\ \mathcal{U}) \to D(\mathcal{S},\mathcal{U})$ has got the functor*

$$\widehat{(-)} : D(\mathcal{S},\mathcal{U}) \to Mod(\mathcal{S},\ \mathcal{U})$$

as left adjoint.

Proof. Note that to prove the triangular identity $(\varepsilon_{\widehat{F}})_x \cdot (\widehat{\eta_F})_x = id_{\widehat{F}_0(x)}$ for any F graph-morphism in $D(\mathcal{S},\mathcal{U})$ and $x \in G_0$, we need to derive for $x \in G_0, v \in V, w \in Tree(F,v), f \in \mathcal{D}_1(\nu(v),x)$

$$\widehat{F}_1(f) \cdot Ap(\widehat{F})(\widetilde{\eta_F}(w)) =_{\widehat{F}_0(x)} [\mathsf{inr} < v, w, f >]$$

To derive a proof of this type we make use of the decidable finiteness of the sketch to be able to perform an induction on the tree w by including the quantification of all the suitable arrows of G_1. The induction on the tree $w \in Tree(F,v)$ is performed with respect to the type

$$C(w) \equiv \Sigma_{l \in List(B)} \ (\overline{\pi_1}(l) =_{List(G_1)} g_0 g_1 ... g_{n_1})$$

where

$$B \equiv \Sigma_{f \in G_1} \ ((\delta_0(f) = \nu(v) \ \& \ \widehat{F}_1(f) \cdot Ap(\widehat{F})(\widetilde{\eta_F}(w)) =_{\widehat{F}_0(x)} [\mathsf{inr} < v, w, f >]) \ \oplus \ \delta_0(f) \neq \nu(v))$$

and $\overline{\pi_1}$ is the lifting of the first projection on lists, and $g_0 g_1 ... g_{n_1}$ is the list of all the arrows in G_1, and $\delta_0(f) \neq \nu(v)$ is defined thanks to the decidability of the sketch. □

Finally as a corollary we get our main theorem:

Corollary 3.16 *Given a finite decidable FP-sketch $\mathcal{S}$, the inclusion functor*

$$I : Mod(\mathcal{S}, \ \mathcal{U}) \to \mathcal{U}^{\mathcal{S}}$$

has got a left adjoint.

Proof. We consider

$$Mod(\mathcal{S}, \ \mathcal{U}) \quad \xhookrightarrow{\mathcal{I}} \quad \mathcal{D}(\mathcal{S},\mathcal{U}) \quad \xhookrightarrow{\mathcal{J}} \quad \mathcal{U}^{\mathcal{S}}$$

and by prop 3.1 and theorem 3.15 the functor $I = \mathcal{J} \cdot \mathcal{I}$ has got a left adjoint.□

4 Conclusions

An analogous reflection can be proved also for finite decidable lex-sketches internal to an arithmetic universe by building more complicated trees. Certainly, we can investigate the existence of the reflection for a wider class of finite product sketches within an arithmetic universe or variations of it. We think that within a locally closed arithmetic universe the reflection should hold for generic internal finite product sketches, since in this paper the property of decidable finiteness of the sketch was essentially used just to quantify over a subset of sketch arrows.

These reflections could be used to build theories of the considered sketches as in [1] but considering an arithmetic universe as our set-theoretic universe in place of the category of Sets. On this point, we also want to explore the applicability of the techniques developed in [7].

In the future we hope to get similar results for finite decidable lex sketches with also coproduct cocones, always by taking, as our set-theoretic universe, an arithmetic universe or some other predicative variations of it. Indeed, we ultimately hope that this kind of results for finite decidable sketches could be useful for applications to database modelling as presented in [6], because working within an intuitionistic predicative universe forces us to perform more effective constructions based on more elementary properties than working within the category of Sets.

Acknowledgement

My acknowledgements go first to Martin Hyland, who proposed me the topic treated here and helped me with many fruitful discussions during my staying in Cambridge. Many thanks also to Steve Vickers, for providing me Gavin Wraith's unpublished notes with the masterthesis of his student [13] to work at his conjecture and to Peter Johnstone for some helpful discussions. Finally, I wish to thank Pino Rosolini and Silvio Valentini for their constant generous promptness in discussing my research work.

References

[1] M. Barr and C. Wells. *Toposes, triples and theories.*, volume 278 of *A Series of Comprehensive Studies in Mathematics*. Springer Verlag, Berlin, 1985.

[2] J.R.B. Cockett. List-arithmetic distributive categories: locoi. *Journal of Pure and Applied Algebra*, 66:1–29, 1990.

[3] N.G. de Bruijn. Telescopic mapping in typed lambda calculus. *Information and Computation*, 91:189–204, 1991.

[4] A. Joyal and I. Moerdijk. *Algebraic set theory*, volume 220 of *Lecture Note Series*. Cambridge University Press, 1995.

[5] P. T. Johnstone. *Sketches of an elephant: a topos theory compendium. Vol. 2.*, volume 43 of *Oxford Logic Guides*. The Clarendon Press, Oxford University Press, New York,, 2002.

[6] M. Johnson, R. Rosebrugh, and R. Wood. Entity relationship attribute designs and sketches. *Theory and Application of Categories*, 10:94–112, 2002.

[7] Y. Kinoshita, J. Power, and M. Takeyama. Sketches. *J. Pure Appl. Algebra*, 143:275–291, 1999. Special volume on the occasion of the 60th birthday of Professor Michael Barr (Montreal, QC, 1997).

[8] M.E. Maietti. Modular correspondence between dependent type theories and categorical universes. *Mittag-Leffler Preprint Series*, 44, 2001.

[9] M.E. Maietti. Joyal's arithmetic universes via type theory. In *Category Theory in Computer Science, 2002*, volume 69 of *Elettronic Notes in Theoretical Computer Science*. Elsevier, 2002.

[10] M.E. Maietti. Joyal's arithmetic universe as list-arithmetic pretopos. See http://www.math.unipd.it/~maietti/pubb.html, 2004.

[11] P. Martin-Löf. *Intuitionistic Type Theory, notes by G. Sambin of a series of lectures given in Padua, June 1980.* Bibliopolis, Naples, 1984.

[12] S. MacLane and I. Moerdijk. *Sheaves in Geometry and Logic. A first introduction to Topos theory.* Springer Verlag, 1992.

[13] A. Morrison. Reasoning in arithmetic universes. Master's thesis, University of London - Imperial College of Science, Technology and Medicine, Advisor: S. Vickers, September 1996.

[14] M. Makkai and G. Reyes. *First order categorical logic.*, volume 611 of *Lecture Notes in Mathematics*. Springer Verlag, 1977.

[15] B. Nordström, K. Petersson, and J. Smith. *Programming in Martin Löf's Type Theory.* Clarendon Press, Oxford, 1990.

[16] A.M. Pitts. Categorical logic. In Oxford University Press, editor, *Handbook of Logic in Computer Science*, volume 5, pages 39–128, 2000.

[17] Th. Streicher. *Semantics of type theory.* Birkhäuser, 1991.

[18] P. Taylor. *Inside every model of Abstract Stone Duality lies an Arithmetic Universe.* In this volume. 2004.

A The internal dependent type theory of arithmetic universes

Here, we recall the description of the typed calculus $\mathcal{A}u$ which provides the internal languages for arithmetic universes, that is list-arithmetic pretoposes, as proved in [8]. The calculus is equipped with types, which should be thought of as sets or data types, and with typed terms which represent proofs of the types to which they belong.
In the style of Martin-Löf's type theory [15], we have four kinds of judgements:

$$A\ type\ [\Gamma] \quad A = B\ [\Gamma] \quad a \in A\ [\Gamma] \quad a = b \in A\ [\Gamma]$$

that is the type judgement, the equality between types, the term judgement and the equality between terms of the same type. The contexts Γ of these judgements are telescopic [3], since types are allowed to depend on variables of other types. The contexts are generated by the following rules

$$1C)\ \ \emptyset\ \ cont \qquad 2C)\ \ \frac{\Gamma\ \ cont \qquad A\ type\ [\Gamma]}{\Gamma, x \in A\ \ cont}\ (x \in A \notin \Gamma)$$

plus the rules of equality between contexts [17], [16]. In the following, we present the inference rules to construct type judgements and term judgements with their equality judgements by recursion. One should also add all the inference rules that express reflexivity, symmetry and transitivity of the equality

between types and terms together with the following set equality rule and assumption of typed variables

$$\textit{set rule)}\quad \frac{a \in A\ [\Gamma] \quad A = B\ [\Gamma]}{a \in B\ [\Gamma]} \qquad \textit{var)}\quad \frac{\Gamma, x \in A, \Delta \quad \textit{cont}}{x \in A\ [\Gamma, x \in A, \Delta]}$$

We can derive then the structural rules of weakening and of a suitable exchange. In the following we give the formation rules for types specific to $\mathcal{A}u$ with the corresponding introduction, elimination and conversion rules of their terms. We omit the equality rules of all the type and term constructors that are necessary to derive the substitution rules. We adopt the usual definitions of bound and free occurrences of variables and we identify two terms under α-conversion. Note that the context common to all judgements involved in a rule will be omitted. The typed variable appearing in a context is meant to be added to the implicit context as the last one. The rules to generate $\mathcal{A}_u$'s types and terms are all present in the extensional version of Martin-Löf's type theory [11] except for the disjointness axiom, the rules about quotients types restricted to *mono* equivalence relations and the effectiveness axiom. A type is called *mono* if it is inhabitated by at most one proof.

Supposing $A\ type$ and $R(x,y)\ type\ [x,y \in A]$, we will write $\mathsf{Equiv}(R)$ to mean the following three judgements: $\mathsf{refl}(x) \in R(x,x)\ [x \in A]$, $\mathsf{sym}(x,y,z) \in R(y,x)\ [x \in A, y \in A, z \in R(x,y)]$, $\mathsf{trans}(x,y,z,u,v) \in R(x,z)\ [x \in A, y \in A, z \in A, u \in R(x,y), v \in R(y,z)]$. Moreover, we will write $\mathsf{Mono}(R)$ to mean

$$z = w \in R(x,y)\ [x \in A, y \in A, z \in R(x,y), w \in R(x,y)]$$

The $\mathcal{A}u$ dependent typed calculus

Terminal type

$$\text{Tr)}\ \top\ type \qquad \text{I-Tr)}\ \star \in \top \qquad \text{C-Tr)}\ \frac{t \in \top}{t = \star \in \top}$$

False type

$$\text{Fs)}\ \bot\ type \qquad \text{E-Fs)}\ \frac{a \in \bot \quad A\ type}{\mathsf{r_o}(a) \in A}$$

Indexed Sum type

$$\Sigma)\ \frac{C(x)\ type \quad [x \in B]}{\Sigma_{x \in B} C(x)\ type} \qquad \text{I-}\Sigma)\ \frac{b \in B \quad c \in C(b)}{< b, c > \in \Sigma_{x \in B} C(x)}$$

$$\text{E-}\Sigma)\ \frac{d \in \Sigma_{x \in B} C(x) \quad m(x,y) \in M(< x, y >)\ [x \in B, y \in C(x)]}{El_\Sigma(d,m) \in M(d)}$$

$$\text{C-}\Sigma)\ \frac{b \in B \quad c \in C(b) \quad m(x,y) \in M(< x, y >)\ [x \in B, y \in C(x)]}{El_\Sigma(< b, c >, m) = m(b,c) \in M(< b, c >)}$$

Equality type

$$\text{Eq)}\ \frac{C\ type \quad c \in C \quad d \in C}{\mathsf{Eq}(C,c,d)\ type} \qquad \text{I-Eq)}\ \frac{c \in C}{\mathsf{eq_C}(c) \in \mathsf{Eq}(C,c,c)}$$

$$\text{E-Eq)}\ \frac{p \in \mathsf{Eq}(C,c,d)}{c = d \in C} \qquad \text{C-Eq)}\ \frac{p \in \mathsf{Eq}(C,c,d)}{p = \mathsf{eq_C}(c) \in \mathsf{Eq}(C,c,d)}$$

Disjoint Sum type

$$+)\ \frac{C\ type \quad B\ type}{C + B\ type} \qquad \text{I}_1\text{-}+)\ \frac{c \in C}{\mathsf{inl}(c) \in C + B} \qquad \text{I}_2\text{-}+)\ \frac{b \in B}{\mathsf{inr}(b) \in C + B}$$

$$\text{E-}+)\ \frac{w \in C + B \quad a_C(x) \in A(\mathsf{inl}(x))\ [x \in C] \quad a_B(y) \in A(\mathsf{inr}(y))\ [y \in B]}{El_+(w, a_C, a_B) \in A(w)}$$

$$\text{C}_1\text{-}+)\ \frac{c \in C \quad a_C(x) \in A(\mathsf{inl}(x))\ [x \in C] \quad a_B(y) \in A(\mathsf{inr}(y))\ [y \in B]}{El_+(\mathsf{inl}(c), a_C, a_B) = a_C(c) \in A(\mathsf{inl}(c))}$$

$$\text{C}_2\text{-}+)\ \frac{b \in B \quad a_C(x) \in A(\mathsf{inl}(x))\ [x \in C] \quad a_B(y) \in A(\mathsf{inr}(y))\ [y \in B]}{El_+(\mathsf{inr}(b), a_C, a_B) = a_B(b) \in A(\mathsf{inr}(b))}$$

Disjointness

$$\frac{c \in C \quad b \in B \quad \mathsf{inl}(c) = \mathsf{inr}(b) \in C + B}{\mathsf{dsj}(c,b) \in \bot}$$

Quotient type

$$\text{Q)}\ \frac{R(x,y)\ type\ [x \in A, y \in A] \quad \mathsf{Mono}(R) \quad \mathsf{Equiv}(R)}{A/R\ type}$$

$$\text{I-Q)}\ \frac{a \in A\ \ A/R\ type}{[a] \in A/R} \qquad \text{eq-}Q)\ \frac{a \in A \quad b \in A \quad d \in R(a,b)\ \ A/R\ type}{[a] = [b] \in A/R}$$

$$\text{E-Q)}\ \frac{p \in A/R \quad l(x) \in L([x])\ [x \in A] \quad l(x) = l(y) \in L([x])\ [x \in A, y \in A, d \in R(x,y)]}{El_Q(l,p) \in L(p)}$$

$$\text{C-Q)}\ \frac{a \in A \quad l(x) \in L([x])\ [x \in A] \quad l(x) = l(y) \in L([x])\ [x \in A, y \in A, d \in R(x,y)]}{El_Q(l,[a]) = l(a) \in L([a])}$$

Effectiveness

$$\frac{a \in A \quad b \in A \quad [a] = [b] \in A/R}{\mathsf{eff}(a,b) \in R(a,b)}$$

List type

$$\text{list)}\ \frac{C\ type}{List(C)\ type} \qquad \text{I}_1\text{-list)}\ \epsilon \in List(C) \qquad \text{I}_2\text{-list)}\ \frac{s \in List(C) \quad c \in C}{\mathsf{cons}(s,c) \in List(C)}$$

$$\text{E-list)}\ \frac{s \in List(C) \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,s) \in L(s)}$$

$$\text{C}_1\text{-list)}\ \frac{s \in List(C) \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,\epsilon) = a \in L(\epsilon)}$$

$$\text{C}_2\text{-list)}\ \frac{s \in List(C) \quad c \in C \quad a \in L(\epsilon) \quad l(x,y,z) \in L(\mathsf{cons}(x,y))\ [x \in List(C), y \in C, z \in L(x)]}{El_{List}(a,l,\mathsf{cons}(s,c)) = l(s,c,El_{List}(a,l,s)) \in L(\mathsf{cons}(s,c))}$$

Note that $List(\top)$ corresponds to the type of natural numbers represented as lists on a singleton. Hence, we put $N \equiv List(\top)$ with $0 \equiv \epsilon$ and $s(n) \equiv \mathsf{cons}(n, *)$ for $n \in List(\top)$.