Inductive Types in Homotopy Type Theory

S. Awodey

March 18, 2014

Abstract

Homotopy type theory is an interpretation of Martin-Löf's constructive type theory into abstract homotopy theory. There results a link between constructive mathematics and algebraic topology, providing topological semantics for intensional systems of type theory as well as a computational approach to algebraic topology via type theory-based proof assistants such as Coq.

The present work investigates inductive types in this setting. Modified rules for inductive types, including types of well-founded trees, or W-types, are presented, and the basic homotopical semantics of such types are determined. Proofs of all results have been formally verified by the Coq proof assistant, and the proof scripts for this verification form an essential component of this research.

Introduction

The constructive type theories introduced by Martin-Löf are dependently-typed λ-calculi with operations for identity types $\mathsf{Id}_A(a,b)$, dependent products $(\Pi x:A)B(x)$ and dependent sums $(\Sigma x:A)B(x)$, among others [22, 23, 24, 28, 29]. These are related to the basic concepts of predicate logic, *viz.* equality and quantification, via the familiar propositions-as-types correspondence [17]. The different systems introduced by Martin-Löf over the years vary greatly both in proof-theoretic strength [14] and computational properties. From the computational point of view, it is important to distinguish between the extensional systems, that have a stronger notion of equality, but for which type-checking is undecidable, and the intensional ones, that have a weaker notion of equality, but for which type-checking is decidable [16]. For example, the type theory presented in [24] is extensional, while that in [29] is intensional.

The difference between the extensional and the intensional treatment of equality has a strong impact also on the properties of the various types that may be assumed in a type theory, and in particular on those of inductive types, such as the types of Booleans, natural numbers, lists and W-types [23]. Within extensional type theories, inductive types can be characterized (up to isomorphism) as initial algebras of certain definable functors. The initiality condition translates directly into a recursion principle that expresses the existence and

uniqueness of recursively-defined functions. In particular, W-types can be characterized as initial algebras of polynomial functors [10, 27]. Furthermore, within extensional type theories, W-types allow us to define a wide range of inductive types, such as the type of natural numbers and types of lists [10, 11, 1]. Within intensional type theories, by contrast, the correspondence between inductive types and initial algebras breaks down, since it is not possible to prove the uniqueness of recursively-defined functions. Furthermore, the reduction of inductive types like the natural numbers to W-types fails [10, 13].

In the present work, we exploit insights derived from the new models of intensional type theory based on homotopy-theoretic ideas [6, 37, 35] to investigate inductive types, thus contributing to the new area known as Homotopy Type Theory. Homotopical intuition justifies the assumption of a limited form of function extensionality, which, as we show, suffices to deduce uniqueness properties of recursively-defined functions up to homotopy. Building on this observation, we introduce the notions of *weak algebra homomorphism* and *homotopy-initial algebra*, which require uniqueness of homomorphisms up to homotopy. We modify the rules for W-types by replacing the definitional equality in the standard computation rule with its propositional counterpart, yielding a weak form of the corresponding inductive type. Our main result is that these new, weak W-types correspond precisely to homotopy-initial algebras of polynomial functors. Furthermore, we indicate how homotopical versions of various inductive types can be defined as special cases of the general construction in the new setting

The work presented here is motivated in part by the Univalent Foundations program formulated by Voevodsky [39]. This ambitious program intends to provide comprehensive foundations for mathematics on the basis of homotopically-motivated type theories, with an associated computational implementation in the Coq proof assistant. The present investigation of inductive types serves as an example of this new paradigm: despite the fact that the intuitive basis lies in higher-dimensional category theory and homotopy theory, the actual development is strictly syntactic, allowing for direct formalization in Coq. Proof scripts of the definitions, results, and all necessary preliminaries are provided in a downloadable repository [5]. An overview of these files is provided as an appendix to this paper.

The paper is organized as follows. In section 1, we describe and motivate the dependent type theory over which we will work and compare it to some other well-known systems in the literature. The basic properties of the system and its homotopical interpretation are developed to the extent required for the present purposes. Section 2 reviews the basic theory of W-types in extensional type theory and sketches the proof that these correspond to initial algebras of polynomial functors; there is nothing new in this section, rather it serves as a framework for the generalization that follows. Section 2 on intensional W-types contains the development of our new theory; it begins with a simple example, that of the type 2 of Boolean truth values, which serves to indicate the main issues involved with inductive types in the intensional setting, and our proposed solution. We then give the general notion of weak W-types, including the crucial new notion of *homotopy-initiality*, and state our main result, the equiv-

alence between the type-theoretic rules for weak W-types and the existence of a homotopy-initial algebra of the corresponding polynomial functor. Moreover, we show how some of the difficulties with intensional W-types are remedied in the new setting by showing that the type of natural numbers can be defined as an appropriate W-type. Finally, we conclude by indicating how this work fits into the larger study of inductive types in Homotopy Type Theory and the Univalent Foundations program generally.

1 Preliminaries

The general topic of Homotopy Type Theory is concerned with the study of the constructive type theories of Martin-Löf under their new interpretation into abstract homotopy theory and higher-dimensional category theory. Martin-Löf type theories are foundational systems which have been used to formalize large parts of constructive mathematics, and also for the development of high-level programming languages [23]. They are prized for their combination of expressive strength and desirable proof-theoretic properties. One aspect of these type theories that has led to special difficulties in providing semantics is the intensional character of equality. In recent work [6, 37, 35, 3], it has emerged that the topological notion of *homotopy* provides an adequate basis for the semantics of intensionality. This extends the paradigm of computability as continuity, familiar from domain theory, beyond the simply-typed λ-calculus to dependently-typed theories involving:

(i) dependent sums $(\Sigma x\colon A)B(x)$ and dependent products $(\Pi x\colon A)B(x)$, modelled respectively by the total space and the space of sections of the fibration modelling the dependency of $B(x)$ over $x : A$;

(ii) and, crucially, including the identity type constructor $\mathsf{Id}_A(a, b)$, interpreted as the space of all *paths* in A between points a and b.

In the present work, we build on this homotopical interpretation to study inductive types, such as the natural numbers, Booleans, lists, and W-types. Within extensional type theories, W-types can be used to provide a constructive counterpart of the classical notion of a well-ordering [24] and to uniformly define a variety of inductive types [10]. However, most programming languages and proof assistants, such as Coq [8], Agda [30] and Epigram [26] use schematic inductive definitions [9, 31] rather than W-types to define inductive types. This is due in part to the practical convenience of the schematic approach, but it is also a matter of necessity; these systems are based on intensional rather than extensional type theories, and in the intensional theory the usual reductions of inductive types to W-types fail [10, 25]. Nonetheless, W-types retain great importance from a theoretical perspective, since they allow us to internalize in type theory arguments about inductive types. Furthermore, as we will see in Section 3, a limited form of extensionality licensed by the homotopical interpretation suffices to develop the theory of W-types in a satisfactory way. In

particular, we shall make use of ideas from higher category theory and homotopy theory to understand W-types as "homotopy-initial" algebras of an appropriate kind.

1.1 Extensional vs. intensional type theories

We work here with type theories that have the four standard forms of judgement

$$A : \mathsf{type}\,, \quad A = B : \mathsf{type}\,, \quad a : A\,, \quad a = b : A\,.$$

We refer to the equality relation in these judgements as *definitional equality*, which should be contrasted with the notion of *propositional equality* recalled below. Such a judgement J can be made also relative to a *context* Γ of variable declarations, a situation that we indicate by writing $\Gamma \vdash J$. When stating deduction rules we make use of standard conventions to simplify the exposition, such as omitting the mention of a context that is common to premisses and conclusions of the rule. The rules for identity types in intensional type theories are given in [29, Section 5.5]. We recall them here in a slighly different, but equivalent, formulation.

- Id-formation rule.
$$\frac{A : \mathsf{type} \quad a : A \quad b : A}{\mathsf{Id}_A(a,b) : \mathsf{type}}$$
- Id-introduction rule.
$$\frac{a : A}{\mathsf{refl}(a) : \mathsf{Id}_A(a,a)}$$
- Id-elimination rule.
$$\frac{\begin{array}{l} x,y : A, u : \mathsf{Id}_A(x,y) \vdash C(x,y,u) : \mathsf{type} \\ x : A \vdash c(x) : C(x,x,\mathsf{refl}(x)) \end{array}}{x,y : A, u : \mathsf{Id}_A(x,y) \vdash \mathsf{idrec}(x,y,u,c) : C(x,y,u)}$$
- Id-computation rule.
$$\frac{\begin{array}{l} x,y : A, u : \mathsf{Id}_A(x,y) \vdash C(x,y,u) : \mathsf{type} \\ x : A \vdash c(x) : C(x,x,\mathsf{refl}(x)) \end{array}}{x : A \vdash \mathsf{idrec}(x,x,\mathsf{refl}(x),c) = c(x) : C(x,x,\mathsf{refl}(x))\,.}$$

As usual, we say that two elements $a, b : A$ are *propositionally equal* if the type $\mathsf{Id}(a,b)$ is inhabited. Most work on W-types to date (*e.g.* [10, 27, 1]) has been in the setting of extensional type theories, in which the following rule, known as the *identity reflection rule*, is also assumed:

$$\frac{p : \mathsf{Id}_A(a,b)}{a = b : A} \tag{1}$$

This rule collapses propositional equality with definitional equality, thus making the overall system somewhat simpler to work with. However, it destroys the constructive character of the intensional system, since it makes type-checking undecidable [16]. For this reason, it is not assumed in the most recent formulations of Martin-Löf type theories [29] or in automated proof assistants like Coq [8].

In intensional type theories, inductive types cannot be characterized by standard category-theoretic universal properties. For instance, in this setting it is not possible to show that there exists a definitionally-unique function out of the empty type with rules as in [29, Section 5.2], thus making it impossible to prove that the empty type provides an initial object. Another consequence of this fact is that, if we attempt to define the type of natural numbers as a W-type in the usual way, then the usual elimination and computation rules for it are no longer derivable [10]. Similarly, it is not possible to show the uniqueness of recursively-defined functions out of W-types. When interpreted categorically, the uniqueness of such functions translates into the initiality property of the associated polynomial functor algebra, which is why the correspondence between W-types and initial algebras fails in the intensional setting.

Due to this sort of poor behaviour of W-types, and other constructions, in the purely intensional setting, that system is often augmented by other extensionality principles that are somewhat weaker than the Reflection rule, such as Streicher's K-rule or the Uniqueness of Identity Proofs (UIP) [34], which has recently been reconsidered in the context of Observational Type Theory [2]. Inductive types in such intermediate systems are somewhat better behaved, but still exhibit some undesirable properties, making them less useful for practical purposes than one might wish [25]. Moreover, these intermediate systems seem to lack a clear conceptual basis: they neither intend to formalize constructive sets (like the extensional theory) nor is there a principled reason to choose these particular extensionality rules, beyond their practical advantages.

1.2 The system $\mathcal{H}$

We here take a different approach to inductive types in the intensional setting, namely, one motivated by the homotopical interpretation. It involves working over a dependent type theory $\mathcal{H}$ which has the following deduction rules on top of the standard structural rules:

- rules for identity types as stated above;
- rules for Σ-types as in [29, Section 5.8];
- rules for Π-types as in [12, Section 3.2];
- the Function Extensionality axiom (FE), *i.e.* the axiom asserting that for every $f, g : A \to B$, the type

$$(\Pi x : A)\mathsf{Id}_B(\mathsf{app}(f, x), \mathsf{app}(g, x)) \to \mathsf{Id}_{A\to B}(f, g)$$

  is inhabited.

Here, we have used the notation $A \to B$ to indicate function types, defined via Π-types in the usual way. Similarly, we will write $A \times B$ to denote the binary product of two types as usually defined via Σ-types.

Remarks

(i) The rules for Π-types of $\mathcal{H}$ are derivable from those in [29, Section 5.4]. For simplicity, we will write $f(a)$ or fa instead of $\mathsf{app}(f, a)$.

(ii) As shown in [38], the principle of propositional function extensionality stated above implies the corresponding principle for dependent functions, *i.e.*

$$(\Pi x : A)\mathsf{Id}_{B(x)}(fx, gx) \to \mathsf{Id}_{(\Pi x:A)B(x)}(f, g)\,.$$

(iii) The following form of the η-rule for Σ-types is derivable:

$$\frac{c : (\Sigma x : A)B(x)}{\eta_\Sigma(c) : \mathsf{Id}(c, \mathsf{pair}(\pi_1 c\,, \pi_2 c))\,,}$$

where π_1 and π_2 are the projections. This can be proved by Σ-elimination, without FE.

(iv) The following form of the η-rule for Π-types is derivable:

$$\frac{f : (\Pi x : A)B(x)}{\eta_\Pi(f) : \mathsf{Id}(f,\ \lambda x.\, fx)}$$

This is an immediate consequence of FE and clearly implies the corresponding η-rule for function types.

(v) $\mathcal{H}$ does *not* include the η-rules as definitional equalities, either for Σ-types or for Π-types (as is done in [13]).

(vi) The type theory $\mathcal{H}$ will serve as the background theory for our study of inductive types and W-types. For this reason, we need not assume it to have any primitive types.

This particular combination of rules is motivated by the fact that $\mathcal{H}$ has a clear homotopy-theoretic sematics. Indeed, the type theory $\mathcal{H}$ is a subsystem of the type theory used in Voevodsky's Univalent Foundations library [38]. In particular, the Function Extensionality axiom is formally implied by Voevodsky's Univalence axiom [37], which is also valid in homotopy-theoretic models, but will not be needed here. Note that, while the Function Extensionality axiom is valid also in set-theoretic models, the Univalence axiom is not. Although $\mathcal{H}$ has a straightforward set-theoretical semantics, we stress that it does not have any global extensionality rules, like the identity reflection rule, K, or UIP. This makes it also compatible with "higher-dimensional" interpretations such as the groupoid model [15], in which the rules of $\mathcal{H}$ are also valid.

1.3 Homotopical semantics

The homotopical semantics of $\mathcal{H}$ is based on the idea that an identity term $p : \mathsf{Id}_A(a, b)$ is (interpreted as) a path $p : a \rightsquigarrow b$ between the points a and b in the space A. More generally, the interpretations of terms $a(x)$ and $b(x)$ with free variables will be continuous functions into the space A, and an identity term $p(x) : \mathsf{Id}_A\big(a(x), b(x)\big)$ is then a continuous family of paths, *i.e.* a homotopy between the continuous functions. Now, the main import of the Id-elimination rule is that type dependency must respect identity, in the following sense: given a dependent type

$$x : A \vdash B(x) : \mathsf{type}\,, \tag{2}$$

and $p : \mathsf{Id}_A(a, b)$, there is then a *transport* function

$$p_! : B(a) \to B(b),$$

which is defined by Id-elimination, taking for $x : A$ the function $\mathsf{refl}(x)_! : B(x) \to B(x)$ to be the identity on $B(x)$. Semantically, given that an identity term $p : \mathsf{Id}_A(a, b)$ is interpreted as a path $p : a \rightsquigarrow b$, this means that a dependent type as in (2) must be interpreted as a space $B \to A$, fibered over the space A, and that the judgement

$$x, y : A \vdash \mathsf{Id}_A(x, y) : \mathsf{type}$$

is interpreted as the canonical fibration $A^I \to A \times A$ of the path space A^I over $A \times A$. For a more detailed overview of the homotopical interpretation, see [3].

Independently of this interpretation, each type A can be shown to carry the structure of a weak ω-groupoid in the sense of [7, 18] with the elements of A as objects, identity proofs $p : \mathsf{Id}_A(a, b)$ as morphisms and elements of iterated identity types as n-cells [36, 19]. Furthermore, $\mathcal{H}$ determines a weak ω-category $\mathcal{C}(\mathcal{H})$ having types as 0-cells, elements $f : A \to B$ as 1-cells, and elements of (iterated) identity types as n-cells [20]. The relation between the weak ω-category structure of $\mathcal{C}(\mathcal{H})$ and the homotopical interpretation of intensional type theories closely mirrors that between higher category theory and homotopy theory in modern algebraic topology, and some methods developed in the latter setting are also applicable in type theory. For instance, the topological notion of contractibility admits the following type-theoretic counterpart, originally introduced by Voevodsky in [38].

Definition 1. A type A is called *contractible* if the type

$$\mathsf{iscontr}(A) =_{\mathrm{def}} (\Sigma x : A)(\Pi y : A)\mathsf{Id}_A(x, y) \tag{3}$$

is inhabited.

The type $\mathsf{iscontr}(A)$ can be seen as the propositions-as-types translation of the formula stating that A has a unique element. However, its homotopical interpretation is as a space that is inhabited if and only if the space interpreting A is contractible in the usual topological sense. The notion of contractibility can be used to articulate the world of types into different homotopical dimensions, or

h-levels [38]. This classification has proven to be quite useful in understanding intensional type theory. For example, it permits the definition of new notions of *proposition* and *set* which provide a useful alternative to the standard approach to formalization of mathematics in type theory [38].

Remark 2. If A is a contractible type, then for every $a, b : A$, the type $\mathsf{Id}_A(a, b)$ is again contractible. This can be proved by Id-elimination [5].

Let us also recall from [38] the notions of weak equivalence and homotopy equivalence. To do this, we need to fix some notation. For $f : A \to B$ and $y : B$, define the type

$$\mathsf{hfiber}(f, y) =_{\mathrm{def}} (\Sigma x : A)\mathsf{Id}_B(fx, y)\,.$$

We refer to this type as the *homotopy fiber* of f at y.

Definition 3. Let $f : A \to B$.

- We say that f is a *weak equivalence* if the type

  $$\mathsf{isweq}(\mathsf{f}) =_{\mathrm{def}} (\Pi y : B)\,\mathsf{iscontr}(\mathsf{hfiber}(f, y))$$

  is inhabited.

- We say that f is a *homotopy equivalence* if there exist a function $g : B \to A$ and elements

  $$\eta : (\Pi x : A)\mathsf{Id}(gfx, x)\,,$$
  $$\varepsilon : (\Pi y : B)\mathsf{Id}(fgy, y)\,.$$

  It is an *adjoint homotopy equivalence* if there are also terms

  $$p : (\Pi x : A)\mathsf{Id}(\varepsilon_{fx}\,, f\,\eta_x)\,,$$
  $$q : (\Pi y : B)\mathsf{Id}(\eta_{gy}\,, g\,\varepsilon_y)\,,$$

  where the same notation for both function application and the action of a function on an identity proof (which is easily definable by Id-elimination), and we write α_x instead of $\alpha(x)$ for better readability.

The type $\mathsf{isweq}(\mathsf{f})$ can be seen as the propositions-as-types translation of the formula asserting that f is bijective, while homotopy equivalence is evidently a form of isomorphism. Thus it is a pleasant fact that a function is a weak equivalence if and only if it is a homotopy equivalence [38]. We also note that all type-theoretic constructions are homotopy invariant, in the sense that they respect this relation of equivalence, a fact which is exploited by the Univalence axiom [37].

In Section 3 below, these and related homotopy-theoretic insights will be used to study inductive types, but first we must briefly review some basic facts about inductive types in the extensional setting.

2 Extensional W-types

We briefly recall the theory of W-types in fully extensional type theories. Let us begin by recalling the rules for W-types from [24]. To state them more conveniently, we sometimes write W instead of $(\mathsf{W}x : A)B(x)$.

- W-formation rule.

$$\frac{A : \mathsf{type} \qquad x : A \vdash B(x) : \mathsf{type}}{(\mathsf{W}x : A)B(x) : \mathsf{type}}$$

- W-introduction rule.

$$\frac{a : A \qquad t : B(a) \to W}{\mathsf{sup}(a, t) : W}$$

- W-elimination rule.

$$\frac{\begin{array}{l} w : W \vdash C(w) : \mathsf{type} \\ x : A\,, u : B(x) \to W\,, v : (\Pi y : B(x))C(u(y)) \vdash \\ \quad c(x, u, v) : C(\mathsf{sup}(x, u)) \end{array}}{w : W \vdash \mathsf{wrec}(w, c) : C(w)}$$

- W-computation rule.

$$\frac{\begin{array}{l} w : W \vdash C(w) : \mathsf{type} \\ x : A\,, u : B(x) \to W\,, v : (\Pi y : B(x))C(u(y)) \vdash \\ \quad c(x, u, v) : C(\mathsf{sup}(x, u)) \end{array}}{\begin{array}{r} x : A, u : B(x) \to W \vdash \mathsf{wrec}(\mathsf{sup}(x, u), c) = \\ c(x, u, \lambda y.\mathsf{wrec}(u(y), c)) : C(\mathsf{sup}(x, u))\,. \end{array}}$$

W-types can be seen informally as the free algebras for signatures with operations of possibly infinite arity, but no equations. Indeed, the premisses of the formation rule above can be thought of as specifying a signature that has the elements of A as operations and in which the arity of $a : A$ is the cardinality of the type $B(a)$. Then, the introduction rule specifies the canonical way of forming an element of the free algebra, and the elimination rule can be seen as the propositions-as-types translation of the appropriate induction principle.

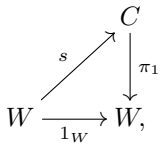
In extensional type theories, this informal description can easily be turned into a precise mathematical characterization. To do so, let us use the theory $\mathcal{H}_{\mathrm{ext}}$ obtained by extending $\mathcal{H}$ with the reflection rule in (1). Let $\mathcal{C}(\mathcal{H}_{\mathrm{ext}})$ be the category with types as objects and elements $f : A \to B$ as maps, in which two maps are considered equal if and only if they are definitionally equal. The premisses of the introduction rule determines the *polynomial endofunctor* $P : \mathcal{C}(\mathcal{H}_{\mathrm{ext}}) \to \mathcal{C}(\mathcal{H}_{\mathrm{ext}})$ defined by

$$P(X) =_{\mathrm{def}} (\Sigma x : A)(B(x) \to X)\,.$$

A P-algebra is a pair consisting of a type C and a function $s_C : PC \to C$, called the structure map of the algebra. The formation rule gives us an object $W =_{\text{def}} (\mathsf{W}x : A)B(x)$ and the introduction rule (in combination with the rules for Π-types and Σ-types) provides a structure map

$$s_W : PW \to W\,.$$

The elimination rule, on the other hand, states that in order for the projection $\pi_1 \colon C \to W$, where $C =_{\text{def}} (\Sigma w : W)C(w)$, to have a section s, as in the diagram



it is sufficient for the type C to have a P-algebra structure over W. Finally, the computation rule states that the section s given by the elimination rule is also a P-algebra homomorphism.

The foregoing elimination rule implies what we call the *simple* W-elimination rule:

$$\frac{C : \mathsf{type} \qquad x : A, v : B(x) \to C \vdash c(x, v) : C}{w : W \vdash \mathsf{simp\text{-}wrec}(w, c) : C}$$

This can be recognized as a recursion principle for maps from W into P-algebras, since the premisses of the rule describe exactly a type C equipped with a structure map $s_C : PC \to C$. For this special case of the elimination rule, the corresponding computation rule again states that the function

$$\lambda w.\mathsf{simp\text{-}wrec}(w, c) : W \to C\,,$$

where $c(x, v) = s_C(\mathsf{pair}(x, v))$ for $x : A$ and $v : B(x) \to C$, is a P-algebra homomorphism. Moreover, this homomorphism can then be shown to be definitionally unique using the elimination rule, the principle of function extensionality and the reflection rule. The converse implication also holds: one can derive the general W-elimination rule from the simple elimination rule and the following η-rule

$$\frac{\begin{array}{l} C : \mathsf{type} \qquad w : W \vdash h(w) : C \\ x : A, v : B(x) \to C \vdash c(x, v) : C \\ x : A\,, u : B(x) \to W \vdash h\,(\mathsf{sup}(x, u)) = c(x, \lambda y.hu(y)) : C \end{array}}{w : W \vdash h(w) = \mathsf{simp\text{-}wrec}(w, c) : C}$$

stating the uniqueness of the $\mathsf{simp\text{-}wrec}$ term among algebra maps. Overall, we therefore have that in $\mathcal{H}_{\text{ext}}$ induction and recursion are interderivable:

Induction	$\Leftrightarrow$	**Recursion**
W-elimination		Simple W-elimination
W-computation		Simple W-computation + η-rule

Finally, observe that what we are calling recursion is equivalent to the statement that the type W, equipped with the structure map $s_W : PW \to W$ is the initial P-algebra. Indeed, assume the simple elimination rule, the simple computation rule and the η-rule; then for any P-algebra $s_C : PC \to C$, there is a function $f : W \to C$ by the simple elimination rule, which is a homomorphism by the computational rule, and is the unique such homomorphism by the η-rule. The converse implication from initiality to recursion is just as direct. Thus, in the extensional theory, to have an initial algebra for the endofunctor P is the same thing as having a type W satisfying the introduction, elimination and computation rules above. Section 3 will be devoted to generalizing this equivalence to the setting of Homotopy Type Theory.

2.1 Inductive types as W-types

To conclude our review, recall that in extensional type theory, many inductive types can be reduced to W-types. We mention the following examples, among many others (see [24], [10], [13], [27], [11], [1]):

1. *Natural numbers.* The usual rules for Nat as an inductive type can be derived from its formalization as the following W-type. Consider the signature with two operations, one of which has arity 0 and one of which has arity 1; it is presented type-theoretically by a dependent type with corresponding polynomial functor (naturally isomorphic to)

   $$P(X) = 1 + X\,,$$

   and the natural numbers Nat together with the canonical element $0 : \mathsf{Nat}$ and the successor function $s : \mathsf{Nat} \to \mathsf{Nat}$ form an initial P-algebra

   $$(0, s) : 1 + \mathsf{Nat} \to \mathsf{Nat}\,.$$

2. *Second number class.* As shown in [24], the second number class can be obtained as a W-type determined by the polynomial functor

   $$P(X) = 1 + X + (\mathsf{Nat} \to X)\,.$$

   This has algebras with three operations, one of arity 0, one of arity 1, and one of arity (the cardinality of) Nat.

3 Intensional W-types

We begin with an example which serves to illustrate, in an especially simple case, some aspects of our theory. The type of Boolean truth values is not a W-type, but it can be formulated as an inductive type in the familiar way by means of formation, introduction, elimination, and computation rules. It then has an "up to homotopy" universal property of the same general kind as the one that we shall formulate in section 3.2 below for W-types, albeit in a simpler form.

3.1 Preliminary example

The standard rules for the type 2 given in [29, Section 5.1] can be stated equivalently as follows.

- 2-formation rule.
$$2 : \mathsf{type}\,.$$
- 2-introduction rules.
$$0 : 2\,, \qquad 1 : 2\,.$$
- 2-elimination rule.
$$\frac{x : 2 \vdash C(x) : \mathsf{type} \qquad c_0 : C(0) \qquad c_1 : C(1)}{x : 2 \vdash \mathsf{2rec}(x, c_0, c_1) : C(x)}$$
- 2-computation rules.
$$\frac{x : 2 \vdash C(x) : \mathsf{type} \qquad c_0 : C(0) \qquad c_1 : C(1)}{\begin{cases} \mathsf{2rec}(0, c_0, c_1) = c_0 : C(0)\,, \\ \mathsf{2rec}(1, c_0, c_1) = c_1 : C(1)\,. \end{cases}}$$

Although these rules are natural ones to consider in the intensional setting, they do not imply a strict universal property. For example, given a type C and elements $c_0, c_1 : C$, the function $\lambda x.\mathsf{2rec}(x, c_0, c_1) : 2 \to C$ cannot be shown to be definitionally unique among the functions $f : 2 \to C$ with the property that $f(0) = c_0 : C$ and $f(1) = c_1 : C$. The best that one can do by using 2-elimination over a suitable identity type, and function extensionality, is to show that it is unique among all such maps up to an identity term, which itself is unique up to a higher identity, which in turn is unique up to .... This sort of weak ω-universality, which apparently involves infinitely much data, can nonetheless be captured directly within the system of type theory (without resorting to coinduction) using ideas from higher category theory. To do so, let us define a 2-*algebra* to be a type C equipped with two elements $c_0\,, c_1 : C$. Then, a *weak homomorphism* of 2-algebras $(f, p_0, p_1) : (C, c_0, c_1) \to (D, d_0, d_1)$ consists of a function $f : C \to D$ together with identity terms

$$p_0 : \mathsf{Id}_D(f(c_0), d_0)\,, \qquad p_1 : \mathsf{Id}_D(f(c_1), d_1)\,.$$

This is a *strict homomorphism* when $f(c_0) = d_0 : D$, $f(c_1) = d_1 : D$ and the identity terms p_0 and p_1 are the corresponding reflexivity terms. We can then define the type of weak homomorphisms from (C, c_0, c_1) to (D, d_0, d_1) by letting

$$\mathsf{2\text{-}Alg}[(C, c_0, c_1), (D, d_0, d_1)] =_{\mathrm{def}} (\Sigma f : C \to D)\mathsf{Id}(f(c_0), d_0) \times \mathsf{Id}_D(f(c_1), d_1)\,.$$

The weak universality condition on the 2-algebra $(2, 0, 1)$ that we seek can now be determined as follows.

Definition 4. A 2-algebra (C, c_0, c_1) is *homotopy-initial* if for any 2-algebra (D, d_0, d_1), the type

$$\mathsf{2\text{-}Alg}\big[(C, c_0, c_1), (D, d_0, d_1)\big]$$

is contractible.

The notion of homotopy initiality, or h-initiality for short, captures in a precise way the informal idea that there is essentially one weak algebra homomorphism $(2, 0, 1) \to (C, c_0, c_1)$. Moreover, h-initiality can be shown to follow from the rules of inference for $\mathsf{2}$ stated above. Indeed, the computation rules for $\mathsf{2}$ stated above evidently make the function

$$\lambda x.\mathsf{2rec}(x, c_0, c_1) : \mathsf{2} \to C$$

into a *strict* algebra map, a stronger condition than is required for h-initiality. Relaxing these definitional equalities to propositional ones, we arrive at the following rules.

- Propositional 2-computation rules.

$$\frac{x : \mathsf{2} \vdash C(x) : \mathsf{type} \qquad c_0 : C(0) \qquad c_1 : C(1)}{\begin{cases} \mathsf{2comp}_0(c_0, c_1) : \mathsf{Id}_{C(0)}\big(\mathsf{2rec}(0, c_0, c_1), c_0\big)\,, \\ \mathsf{2comp}_1(c_0, c_1) : \mathsf{Id}_{C(1)}\big(\mathsf{2rec}(1, c_0, c_1), c_1\big)\,. \end{cases}}$$

This variant is not only still sufficient for h-initiality, but also necessary, as we state precisely in the following.

Proposition 5. *Over the type theory $\mathcal{H}$, the formation, introduction, elimination, and propositional computation rules for* $\mathsf{2}$ *are equivalent to the existence of a homotopy-initial* 2*-algebra.*

Proof sketch. Suppose we have a type $\mathsf{2}$ satisfying the stated rules. Then clearly $(2, 0, 1)$ is a 2-algebra; to show that it is h-initial, take any 2-algebra (C, c_0, c_1). By elimination with respect to the constant family C and the elements c_0 and c_1, we have the map $\lambda x.\mathsf{2rec}(x, c_0, c_1) : \mathsf{2} \to C$, which is a weak algebra homomorphism by the propositional computation rules. Thus we obtain a term $h : \mathsf{2\text{-}Alg}\big[(2, 0, 1), (C, c_0, c_1)\big]$. Now given any $k : \mathsf{2\text{-}Alg}\big[(2, 0, 1), (C, c_0, c_1)\big]$, we need a term of type $\mathsf{Id}(h, k)$. This term follows from a propositional η-rule, which is derivable by $\mathsf{2}$-elimination over a suitable identity type.

Conversely, let $(2, 0, 1)$ be an h-initial 2-algebra. To prove elimination, let $x : \mathsf{2} \vdash C(x) : \mathsf{type}$ with $c_0 : C(0)$ and $c_1 : C(1)$ be given, and consider the 2-algebra (C', c_0', c_1') defined by:

$$\begin{aligned} C' &=_{\mathrm{def}} (\Sigma x : \mathsf{2})C(x)\,, \\ c_0' &=_{\mathrm{def}} \mathsf{pair}(0, c_0)\,, \\ c_1' &=_{\mathrm{def}} \mathsf{pair}(1, c_1)\,. \end{aligned}$$

Since $\mathsf{2}$ is h-initial, there is a map $r : \mathsf{2} \to C'$ with identities $p_0 : \mathsf{Id}(r0, c'_0)$ and $p_1 : \mathsf{Id}(r1, c'_1)$. Now, we would like to set

$$\mathsf{2rec}(x, c_0, c_1) = \pi_2(rx) : C(x),$$

where π_2 is the second projection from $C' = (\Sigma x : \mathsf{2})C(x)$. But recall that in general $\pi_2(z) : C(\pi_1(z))$, and so (taking the case $x = 0$) we have $\pi_2(r0) : C(\pi_1(r0))$ rather than the required $\pi_2(r0) : C(0)$; that is, since it need not be that $\pi_1(r0) = 0$, the term $\pi_2(r0)$ has the wrong type to be $\mathsf{2rec}(0, c_0, c_1)$. However, we can show that

$$\pi_1 : (\Sigma x : \mathsf{2})C(x) \to \mathsf{2}$$

is a weak homomorphism, so that the composite $\pi_1 \circ r : (2, 0, 1) \to (2, 0, 1)$ must be propositionally equal to the identity homomorphism $1_2 : (2, 0, 1) \to (2, 0, 1)$, by the contractibility of $\mathsf{2\text{-}Alg}\big[(2, 0, 1), (2, 0, 1)\big]$. Thus there is an identity term $p : \mathsf{Id}(\pi_1 \circ r, 1_2)$, along which we can transport using $p_! : C(\pi_1(r0)) \to C(0)$, thus taking $\pi_2(r0) : C(\pi_1(r0))$ to the term $p_!(\pi_2(r0)) : C(0)$ of the correct type. We can then set

$$\mathsf{2rec}(x, c_0, c_1) = p_!(\pi_2(rx)) : C(x)$$

to get the required elimination term. The computation rules follow by a rather lengthy calculation. □

Proposition 5 is the analogue in Homotopy Type Theory of the characterization of $\mathsf{2}$ as a strict coproduct $1 + 1$ in extensional type theory. It makes precise the rough idea that, in intensional type theory, $\mathsf{2}$ is a kind of homotopy coproduct or weak ω-coproduct in the weak ω-category $\mathcal{C}(\mathcal{H})$ of types, terms, identity terms, higher identity terms, ... It is worth emphasizing that h-initiality is a purely type-theoretic notion; despite having an obvious semantic interpretation, it is formulated in terms of inhabitation of specific, definable types. Indeed, Proposition 5 and its proof have been completely formalized in the Coq proof assistant [5].

Remark 6. A development entirely analogous to the foregoing can be given for the type Nat of natural numbers. In somewhat more detail, one introduces the notions of a Nat-algebra and of a weak homomorphism of Nat-algebras. Using these, it is possible to define the notion of a homotopy-initial Nat-algebra, analogue to that of a homotopy-initial $\mathsf{2}$-algebra in Definition 4. With these definitions in place, one can prove an equivalence between the formation, introduction, elimination and propositional computation rules for Nat and the existence of a homotopy-initial Nat-algebra. Here, the propositional computation rules are formulated like those above, *i.e.* by replacing the definitional equalities in the conclusion of the usual computation rules [29, Section 5.3] with propositional equalities. We do not pursue this further here, however, since Nat can also be presented as a W-type, as we discuss in section 3.3 below.

3.2 The main theorem

Although it is more elaborate to state (and difficult to prove) owing to the presence of recursively generated data, our main result on W-types is analogous to the foregoing example in the following respect: rather than being strict initial algebras, as in the extensional case, weak W-types are instead homotopy-initial algebras. This fact can again be stated entirely syntactically, as an equivalence between two sets of rules: the formation, introduction, elimination, and propositional computation rules (which we spell out below) for W-types, and the existence of an h-initial algebra, in the appropriate sense. Moreover, as in the simple case of the type 2, the proof of the equivalence is again entirely constructive.

The required definitions in the current setting are as follows. Let us assume that

$$x : A \vdash B(x) : \mathsf{type}\,,$$

and define the associated polynomial functor as before:

$$PX = (\Sigma x : A)(B(x) \to X)\,. \tag{4}$$

(Actually, this is now functorial only up to propositional equality, but this change makes no difference in what follows.) By definition, a P-algebra is a type C equipped a function $s_C : PC \to C$. For P-algebras (C, s_C) and (D, s_D), a *weak homomorphism* between them $(f, s_f) : (C, s_C) \to (D, s_D)$ consists of a function $f : C \to D$ and an identity proof

$$s_f : \mathsf{Id}_{PC \to D}\big(f \circ s_C\,, s_D \circ Pf\big)\,,$$

where $Pf : PC \to PD$ is the result of the easily-definable action of P on $f : C \to D$. Such an algebra homomorphism can be represented suggestively in the form:

$$\begin{array}{ccc} PC & \xrightarrow{Pf} & PD \\ {\scriptstyle s_C}\downarrow & {\scriptstyle s_f} & \downarrow{\scriptstyle s_D} \\ C & \xrightarrow[f]{} & D \end{array}$$

Accordingly, the type of weak algebra maps is defined by

$$P\text{-}\mathsf{Alg}\big[(C, s_C), (D, s_D)\big] =_{\mathrm{def}} \\ (\Sigma f : C \to D)\,\mathsf{Id}(f \circ s_C, s_D \circ Pf)\,.$$

Definition 7. A P-algebra (C, s_C) is *homotopy-initial* if for every P-algebra (D, s_D), the type

$$P\text{-}\mathsf{Alg}\big[(C, s_C), (D, s_D)\big]$$

of weak algebra maps is contractible.

Remark 8. The notion of h-initiality captures a universal property in which the usual conditions of existence and uniqueness are replaced by conditions of existence and uniqueness up to a system of higher and higher identity proofs. To explain this, let us fix a P-algebra (C, s_C) and assume that it is homotopy-initial. Then, given any P-algebra (D, s_D), there is a weak homomorphism $(f, s_f) : (C, s_C) \to (D, s_D)$, since the type of weak maps from (C, s_C) to (D, s_D), being contractible, is inhabited. Furthermore, for any weak map $(g, s_g) : (C, s_C) \to (D, s_D)$, the contractibility of the type of weak maps implies that there is an identity proof

$$p : \mathsf{Id}\big((f, s_f), (g, s_g)\big)\,,$$

witnessing the uniqueness up to propositional equality of the homomorphism (f, s_f). But it is also possible to prove that the identity proof p is unique up to propositional equality. Indeed, since (f, s_f) and (g, s_g) are elements of a contractible type, the identity type $\mathsf{Id}((f, s_f), (g, s_g))$ is also contractible, as observed in Remark 2. Thus, if we have another identity proof $q : \mathsf{Id}((f, s_f), (g, s_g))$, there will be an identity term $\alpha : \mathsf{Id}(p, q)$, which is again essentially unique, and so on. It should also be pointed out that, just as strictly initial algebras are unique up to isomorphism, h-initial algebras are unique up to weak equivalence. It then follows from the Univalence axiom that two h-initial algebras are propositionally equal, a fact that we mention only by the way. Finally, we note that there is also a homotopical version of Lambek's Lemma, asserting that the structure map of an h-initial algebra is itself a weak equivalence, making the algebra a *homotopy fixed point* of the associated polynomial functor. The reader can work out the details from the usual proof and the definition of h-initiality, or consult [5].

The deduction rules that characterize homotopy-initial algebras are obtained from the formation, introduction, elimination and computation rules for W-types stated in Section 2 by simply replacing the W-computation rule with the following rule, that we call the propositional W-computation rule.

- Propositional W-computation rule.

$$\frac{\begin{array}{l} w : W \vdash C(w) : \mathsf{type} \\ x : A, u : B(x) \to W, v : (\Pi y : B(x))C(u(y)) \vdash \\ \qquad c(x, u, v) : C(\mathsf{sup}(x, u)) \end{array}}{\begin{array}{l} x : A, u : B(x) \to W \vdash \mathsf{wcomp}(x, u, c) : \\ \qquad \mathsf{Id}\big(\mathsf{wrec}(\mathsf{sup}(x, u), c), c(x, u, \lambda y.\mathsf{wrec}(u(y), c)\big) \end{array}}$$

Remark 9. One interesting aspect of this group of rules, to which we shall refer as the *rules for homotopical W-types*, is that, unlike the standard rules for W-types, they are invariant under propositional equality. To explain this more precisely, let us work in a type theory with a type universe U closed under all the forms of types of $\mathcal{H}$ and W-types. Let $A : \mathsf{U}$, $B : A \to \mathsf{U}$ and define $W =_{\mathrm{def}} (\mathsf{W}x : A)B(x)$. The invariance of the rules for homotopy W-types under

propositional equality can now be expressed by saying that if we have a type $W' : \mathsf{U}$ and an identity proof $p : \mathsf{Id}_U(W, W')$, then the Id-elimination rule implies that W' satisfies the same rules as W, in the sense that there are definable terms playing the role of the primitive constants that appear in the rules for W.

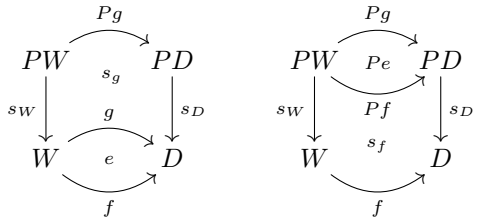
We can now state our main result. Its proof has been formalized in the Coq system, and the proof scripts are available at [5]; thus we provide only an outline of the proof.

Theorem 10. *Over the type theory $\mathcal{H}$, the rules for homotopical W-types are equivalent to the existence of homotopy-initial algebras for polynomial functors.*

Proof sketch. The two implications are proved separately. First, we show that the rules for homotopical W-types imply the existence of homotopy-initial algebras for polynomial functors. Let us assume that $x : A \vdash B(x) : \mathsf{type}$ and consider the associated polynomial functor P, defined as in (4). Using the W-formation rule, we define $W =_{\mathrm{def}} (\mathsf{W}x : A)B(x)$ and using the W-introduction rule we define a structure map $s_W : PW \to W$, exactly as in the extensional theory. We claim that the algebra (W, s_W) is h-initial. So, let us consider another algebra (C, s_C) and prove that the type T of weak homomorphisms from (W, s_W) to (C, s_C) is contractible. To do so, observe that the W-elimination rule and the propositional W-computation rule allow us to define a weak homomorphism $(f, s_f) : (W, s_W) \to (C, s_C)$, thus showing that T is inhabited. Finally, it is necessary to show that for every weak homomorphism $(g, s_g) : (W, s_W) \to (C, s_C)$, there is an identity proof

$$p : \mathsf{Id}((f, s_f), (g, s_g))\,. \tag{5}$$

This uses the fact that, in general, a type of the form $\mathsf{Id}((f, s_f), (g, s_g))$, is weakly equivalent to the type of what we call *algebra 2-cells*, whose canonical elements are pairs of the form (e, s_e), where $e : \mathsf{Id}(f, g)$ and s_e is a higher identity proof witnessing the propositional equality between the identity proofs represented by the following pasting diagrams:



In light of this fact, to prove that there exists a term as in (5), it is sufficient to show that there is an algebra 2-cell

$$(e, s_e) : (f, s_f) \Rightarrow (g, s_g)\,.$$

The identity proof $e : \mathsf{Id}(f, g)$ is now constructed by function extensionality and W-elimination so as to guarantee the existence of the required identity proof s_e.

For the converse implication, let us assume that the polynomial functor associated to the judgement $x : A \vdash B(x) : \mathsf{type}$ has an h-initial algebra (W, s_W). To derive the W-formation rule, we let $(\mathsf{W}x : A)B(x) =_{\mathrm{def}} W$. The W-introduction rule is equally simple to derive; namely, for $a : A$ and $t \colon B(a) \to W$, we define $\mathsf{sup}(a, t) : W$ as the result of applying the structure map $s_W \colon PW \to W$ to $\mathsf{pair}(a, t) : PW$. For the W-elimination rule, let us assume its premisses and in particular that $w : W \vdash C(w) : \mathsf{type}$. Using the other premisses, one shows that the type $C =_{\mathrm{def}} (\Sigma w : W)C(w)$ can be equipped with a structure map $s_C : PC \to C$. By the h-initiality of W, we obtain a weak homomorphism $(f, s_f) : (W, s_W) \to (C, s_C)$. Furthermore, the first projection $\pi_1 : C \to W$ can be equipped with the structure of a weak homomorphism, so that we obtain a diagram of the form

$$\begin{array}{ccccc} PW & \xrightarrow{Pf} & PC & \xrightarrow{P\pi_1} & PW \\ \downarrow{\scriptstyle s_W} & & \downarrow{\scriptstyle s_C} & & \downarrow{\scriptstyle s_W} \\ W & \xrightarrow[f]{} & C & \xrightarrow[\pi_1]{} & W\,. \end{array}$$

But the identity function $1_W : W \to W$ has a canonical structure of a weak algebra homomorphism and so, by the contractibility of the type of weak homomorphisms from (W, s_W) to itself, there must be an identity proof between the composite of (f, s_f) with (π_1, s_{π_1}) and $(1_W, s_{1_W})$. This implies, in particular, that there is an identity proof $p : \mathsf{Id}(\pi_1 \circ f, 1_W)$. Since $(\pi_2 \circ f)w : C((\pi_1 \circ f)w)$, we can define

$$\mathsf{wrec}(w, c) =_{\mathrm{def}} p_!\,((\pi_2 \circ f)w) : C(w)$$

where the transport $p_!$ is defined via Id-elimination over the dependent type

$$u : W \to W \vdash C(u(w)) : \mathsf{type}\,.$$

The verification of the propositional W-computation rule is a rather long calculation, involving several lemmas concerning the naturality properties of operations of the form $p_!$. □

3.3 Definability of inductive types

We conclude this section by indicating how the limited form of extensionality that is assumed in the type theory $\mathcal{H}$, namely the principle of function extensionality, allows us to overcome the obstacles in defining various inductive types as W-types mentioned at the end of Section 2, provided that both are understood in the appropriate homotopical way, *i.e.* with all types being formulated with propositional computation rules.

Consider first the paradigmatic case of the type of natural numbers. To define it as a W-type, we work in an extension of the type theory $\mathcal{H}$ with

- formation, introduction, elimination and propositional computation rules for types $\mathsf{0}$, $\mathsf{1}$ and $\mathsf{2}$ that have zero, one and two canonical elements, respectively;

- the rules for homotopy W-types, as stated above;
- rules for a type universe U reflecting all the forms of types of $\mathcal{H}$, W-types, and $\mathsf{0}$, $\mathsf{1}$ and $\mathsf{2}$.

In particular, the rules for $\mathsf{2}$ are those given in Section 3.1. We then proceed as follows. We begin by setting $A = \mathsf{2}$, as in the extensional case. We then define a dependent type

$$x : A \vdash B(x) : \mathsf{U}$$

by $\mathsf{2}$-elimination, so that the propositional $\mathsf{2}$-computation rules give us propositional equalities

$$p_0 : \mathsf{Id}_U(\mathsf{0}, B(0))\,, \qquad p_1 : \mathsf{Id}_U(\mathsf{1}, B(1))\,.$$

Because of the invariance of the rules for $\mathsf{0}$ and $\mathsf{1}$ under propositional equalities (as observed in Remark 9), we can then derive that the types $B(0)$ and $B(1)$ satisfy rules analogous to those for $\mathsf{0}$ and $\mathsf{1}$, respectively. This allows us to show that the type

$$\mathsf{Nat} =_{\mathrm{def}} (\mathsf{W}x : A)B(x)$$

satisfies the introduction, elimination and propositional computation rules for the type of natural numbers. The proof of this fact proceeds essentially as one would expect, but to derive the propositional computation rules it is useful to observe that for every type $X : \mathsf{U}$, there are adjoint homotopy equivalences, in the sense of Definition 3, between the types $\mathsf{0} \to X$ and $\mathsf{1}$, and between $\mathsf{1} \to X$ and X. Indeed, the propositional identities witnessing the triangular laws are useful in the verification of the propositional computation rules for Nat. For details, see the formal development in Coq provided in [5]. Observe that as a W-type, Nat is therefore also an h-initial algebra for the equivalent polynomial functor $P(X) = \mathsf{1} + X$, as expected.

Finally, let us observe that the definition of a type representing the second number class as a W-type, as discussed in [24], carries over equally well. Indeed, one now must represent type-theoretically a signature with three operations: the first of arity zero, the second of arity one, and the third of arity Nat. For the first two we can proceed exactly as before, while for the third there is no need to prove auxiliary results on adjoint homotopy equivalences. As before, the second number class supports an h-initial algebra structure for the corresponding polynomial functor $P(X) = \mathsf{1} + X + (\mathsf{Nat} \to X)$. Again, the formal development of this result in Coq can be found in [5].

4 Future work

The treatment of W-types presented here is part of a larger investigation of general inductive types in Homotopy Type Theory. We sketch the projected course of our further research.

1. In the setting of extensional type theory, Dybjer [10] showed that every strictly positive definable functor can be represented as a polynomial functor, so that all such inductive types are in fact W-types. This result should generalize to the present setting in a straightforward way.

2. Also in the extensional setting, Gambino and Hyland [11] showed that general tree types [32] [28, Chapter 16], viewed as initial algebras for general polynomial functors, can be constructed from W-types in locally cartesian closed categories, using equalizers. We expect this result to carry over to the present setting as well, using Id-types in place of equalizers.

3. In [37] Voevodsky has shown that all inductive types of the Predicative Calculus of Inductive Constructions can be reduced to the following special cases:

   - $\mathsf{0}$, $\mathsf{1}$, $A + B$, $(\Sigma x : A)B(x)$,
   - $\mathsf{Id}_A(a, b)$,
   - general tree types.

   Combining this with the foregoing, we expect to be able to extend our Theorem 10 to the full system of predicative inductive types underlying Coq.

Finally, one of the most exciting recent developments in Univalent Foundations is the idea of Higher Inductive Types (HITs), which can also involve identity terms in their signature [21, 33]. This allows for algebras with equations between terms, like associative laws, coherence laws, etc.; but the really exciting aspect of HITs comes from the homotopical interpretation of identity terms as paths. Viewed thus, HITs should permit direct formalization of many basic geometric spaces and constructions, such as the unit interval I; the spheres S^n, tori, and cell complexes; truncations, such as the [bracket] types [4]; various kinds of quotient types; homotopy (co)limits; and many more fundamental and fascinating objects of geometry not previously captured by type-theoretic formalizations. Our investigation of conventional inductive types in the homotopical setting should lead to a deeper understanding of these new and important geometric analogues.

Acknowledgements

We would like to thank Vladimir Voevodsky and Michael Warren for helpful discussions on the subject of this paper. In particular, Vladimir Voevodsky suggested a simplification of the proof that the rules for homotopical W-types imply h-initiality.

Steve Awodey gratefully acknowledges the support of the National Science Foundation, Grant DMS-1001191 and the Air Force OSR, Grant 11NL035. Nicola Gambino is grateful for the support of the Institute for Advanced Study,

where he worked on this project. This work was supported by the National Science Foundation under agreement No. DMS-0635607. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation. Kristina Sojakova is grateful for the support of CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and W911NF-09-1-0273 from the Army Research Office.

References

[1] M. Abbott, T. Altenkirch, and N. Ghani. Containers: Constructing strictly positive types. *Th. Comp. Sci.*, 342(1):3–27, 2005.

[2] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In *PLPV '07: Proceedings of the 2007 workshop on Programming languages meets program verification*, pages 57–68. ACM, 2007.

[3] S. Awodey. Type theory and homotopy, 2010. Available from the author's web page.

[4] S. Awodey and A. Bauer. Propositions as [types]. *Journal of Logic and Computation*, 14(4):447–471, 2004.

[5] S. Awodey, N. Gambino, and K. Sojakova. Inductive types in Homotopy Type Theory: Coq proofs, 2011. Available at: www.andrew.cmu.edu/∼awodey/hott/ithottCoq.zip.

[6] S. Awodey and M. A. Warren. Homotopy theoretic models of identity types. *Mathematical Proceedings of the Cambridge Philosophical Society*, 146:45–55, 2009.

[7] M. Batanin. Monoidal globular categories as a natural environment for the theory of weak n-categories. *Advances in Mathematics*, 136(1):39–103, 1998.

[8] Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development. Coq'Art: the Calculus of Inductive Constructions*. Springer Verlag, 2004.

[9] T. Coquand and C. Paulin-Mohring. Inductively defined types. In *Proceedings of Colog'88*, volume 417 of *LNCS*. Springer, 1990.

[10] P. Dybjer. Representing inductively defined sets by wellorderings in Martin-Löf's type theory. *Theor. Comp. Sci.*, 176:329–335, 1997.

[11] N. Gambino and M. Hyland. Wellfounded Trees and Dependent Polynomial Functors. In S. Berardi, M. Coppo, and F. Damiani, editors, *Types for Proofs and Programs (TYPES 2003)*, volume 3085 of *LNCS*, pages 210–225, 2004.

[12] R. Garner. On the strength of dependent products in the type theory of Martin-Löf. *Ann. Pure Appl. Logic*, 160:1–12, 2009.

[13] H. Goguen and Z. Luo. Inductive data types: well-ordering types revisited. In G. Huet and G. Plotkin, editors, *Logical Environments*, pages 198–218. Cambridge University Press, 1993.

[14] E. Griffor and M. Rathjen. The strength of some Martin-Löf type theories. *Arch. Math. Logic*, 33(5):347–385, 1994.

[15] M. Hofmann and T. Streicher. The groupoid model of type theory. In G. Sambin and J. Smith, editors, *Twenty-five years of constructive type theory*. Oxford University Press, 1995.

[16] Martin Hofmann. *Extensional constructs in intensional type theory*. Springer-Verlag, 1997.

[17] W. H. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 479–490. Academic Press, 1980.

[18] T. Leinster. *Higher operads, higher categories*. Cambridge University Press, 2004.

[19] P. Lumsdaine. Weak ω-categories from intensional type theory. In P.-L. Curien, editor, *Typed Lambda Calculi and Applications*, number 5608 in LNCS, pages 172–187. Springer, 2009.

[20] P. Lumsdaine. *Higher categories from type theories*. PhD thesis, Carnegie Mellon University, 2010.

[21] P. Lumsdaine. Higher inductive types: a tour of the managerie, 2011. Post on the Homotopy Type Theory blog.

[22] P. Martin-Löf. An Intuitionistic Theory of Types: Predicative Part. In H. Rose and J. Shepherdson, editors, *Logic Colloquium 1973*, pages 73–118. North-Holland, 1975.

[23] P. Martin-Löf. Constructive mathematics and computer programming. In *Proceedings of the Sixth International Congress for Logic, Methodology and Philosophy of Science*, pages 153–175. North-Holland, 1982.

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

[25] C. McBride. W-types: good news and bad news, 2010. Post on the Epigram blog.

[26] C. McBride and J. McKinna. The view from the left. *Journal of Functional Programming*, 14(1):16–111, 2004.

[27] I. Moerdijk and E. Palmgren. Wellfounded trees in categories. *Annals of Pure and Applied Logic*, 104:189–218, 2000.

[28] B. Nordstrom, K. Petersson, and J. Smith. *Programming in Martin-Löf type theory*. Oxford University Press, 1990.

[29] B. Nordstrom, K. Petersson, and J. Smith. Martin-Löf type theory. In *Handbook of Logic in Computer Science*, volume 5, pages 1–37. Oxford University Press, 2000.

[30] U. Norell. *Towards a Practical Programming Language Based on Dependent Type Theory*. PhD thesis, Chalmers University of Technology, 2007.

[31] C. Paulin-Mohring. Inductive definitions in the system Coq - Rules and Properties. In *Typed Lambda Calculi and Applications*, volume 664 of *LNCS*. Springer, 1993.

[32] K. Petersson and D. Synek. A set constructor for inductive sets in Martin-Löf type theory. In *Proceedings of the 1989 Conference on Category Theory and Computer Science, Manchester, U.K.*, volume 389 of *LNCS*. Springer-Verlag, 1989.

[33] M. Shulman. Homotopy Type Theory, VI, 2011. Post on the n-category café blog.

[34] T. Streicher. Investigations into intensional type theory, 1993. Habilitation Thesis. Available from the author's web page.

[35] B. van den Berg and R. Garner. Topological and simplicial models of identity types, 2011. arXiv:1007.4638v2. To appear in *ACM Transactions in Computational Logic*.

[36] B. van den Berg and R. Garner. Types are weak ω-groupoids. *Proceedings of the London Mathematical Society*, 102(3):370–394, 2011.

[37] V. Voevodsky. Notes on type systems, 2009. Available from the author's web page.

[38] V. Voevodsky. Univalent foundations Coq files, 2010. Available from the author's web page.

[39] V. Voevodsky. Univalent foundations project, 2010. Available from the author's web page.

Appendix: The Accompanying Coq Files

The formal verifications of the results referred to in the text form an integral part of the present research. They are available in the form of a downloadable repository, located at:

`https://github.com/HoTT/HoTT/tree/master/Coq/IT`

These scripts are for use with the Coq system, which is based on the Calculus of Inductive Constructions, a strict extension of the type theory $\mathcal{H}$ considered here. Inspection of the code, however, reveals that all proofs can indeed be carried out within the system $\mathcal{H}$. The README file from the repository is reproduced below for the reader's convenience.

Inductive types in Homotopy Type Theory: Coq proofs

This repository IT contains Coq proofs formalizing the development of inductive types in the setting of Homotopy Type Theory. The files in the folder "`univalent_foundations`" are by V. Voevodsky. All other files have been created and are maintained by S. Awodey, N. Gambino, and K. Sojakova (awodey@cmu.edu, ngambino@math.unipa.it, kristinas@cmu.edu).

The Coq version used is 8.3pl3.

The main results formalized in the repository are the proofs of the following statements:

- weak 2-types arise as h-initial algebras
- weak W-types arise as h-initial algebras
- weak natural numbers reduce to weak W-types
- second number class reduces to weak W-types

The organization is as follows:

1) The folder "`univalent_foundations`" contains the up-to-date development of Voevodsky's Univalent Foundations program, which aims to provide computational foundations for mathematics based on homotopically-motivated type theories. For our purposes only the following files will be needed: – "Generalities/uuu.v" – "Generalities/uu0.v" The above files introduce the identity types, Sigma types, and the (simple) function extensionality axiom. Dependent function extensionality is derived as a consequence and a homotopy equivalence is established between the types of pointwise vs global function equalities.

2) The file "identity.v" in the folder "identity" introduces various lemmas concerning the basic homotopical properties of propositional equalities and the interaction of identity types with other type constructors.

3) The folder "`inductive_types`" contains the definitions of various weak inductive types, namely the types Zero, One, Two, Three with 0,1,2,3 constructors respectively; the type Sum of weak sums; weak natural numbers and lists; and weak W-types. The types are presented in the standard form by giving the formation, introduction, elimination, and computation rules (here called beta). The corresponding eta rules are derived.

4) The folder "two_is_hinitial" contains the proof that the type Two arises as an h-initial algebra. The proof is structured as follows:

   i) First the analogous simple rules for Two are formulated; the corresponding eta rules are no longer derivable and are stated as axioms. This is done in the file "two_simp.v".

   ii) We show that the dependent rules for Two imply the simple ones and vice versa. This is done in the files "simp_implies_dep.v" and "dep_implies_simp.v".

   iii) The notions of algebra homomorphisms, algebra 2-cells, and homotopy-initial algebras are formulated for Two. It is furthermore shown that two algebra homomorphisms are propositionally equal if and only if there exists an algebra 2-cell between them. This is done in the file "two_algebras.v". iv) We show that the simple rules for Two are equivalent to the assertion that there exists a homotopy-initial 2-algebra. This is done in the files "simp_implies_hinitial.v" and "hinitial_implies_simp.v".

5) The folder "w_is_hinitial" contains the proof that weak W-types arise as h-initial algebras for polynomial functors. The proof is structured as follows:

   i) First we introduce the notion of polynomial functors and prove a number of useful lemmas. This is done in the file "polynomial_functors.v".

   ii) We show the main result, i.e. that the dependent rules for W are equivalent to the assertion that there exists a homotopy-initial algebra for the associated polynomial functor. This is done in the files "w_implies_hinitial.v" and "hinitial_implies_w.v".

6) The file "nat_as_w_type.v" in the folder "nat_as_w_type" formalizes the proof that weak natural numbers are encodable as weak W-types in the presence of the types Zero, One, and (a type-level version of) Two.

7) The file "o2_as_w_type.v" in the folder "nat_as_w_type" formalizes the proof that the second number class is encodable as a weak W-type in the presence of the types Zero, One, Two, and (a type-level version of) Three.

Order of compilation:

1) univalent_foundations:

   1.1) Generalities/uuu.v

   2.1) Generalities/uu0.v

2) identity/identity.v

3) inductive_types/*.v

4) two_is_hinitial:

4.1) two_simp.v,two_algebras.v

4.2) dep_implies_simp.v,simp_implies_dep.v, simp_implies_hinitial.v, hinitial_implies_simp.v

5) w_is_hinitial:

5.1) polynomial_functors.v

5.2) hinitial_implies_w.v, w_implies_hinitial.v

6) nat_as_w_type:

6.1) nat_as_w_type.v

6.2) o2_as_w_type.v

Repository last updated : 17 Jan 2012.