

A topological counterpart of well-founded trees in dependent type theory

Maria Emilia Maietti^{a,1} Pietro Sabelli^{a,2}

^a *Department of Mathematics “Tullio Levi-Civita”
University of Padua
Padua, Italy*

Abstract

Within dependent type theory, we provide a topological counterpart of well-founded trees (for short, W-types) by using a proof-relevant version of the notion of inductively generated suplattices introduced in the context of formal topology under the name of “inductively generated basic covers”. In more detail, we show, firstly, that in Homotopy Type Theory, W-types and proof-relevant inductively generated basic covers are propositionally mutually encodable. Secondly, we prove they are definitionally mutually encodable in the Agda implementation of intensional Martin-Löf’s type theory. Finally, we reframe the equivalence in the Minimalist Foundation framework by introducing well-founded predicates as the logical counterpart for predicates of dependent W-types. All the results have been checked in the Agda proof-assistant.

Keywords: dependent type theory, formal topology, well-founded trees, W types, intensional type theory, homotopy type theory.

1 Introduction

It is well known that dependent type theories including Martin-Löf’s type theory in [16] provide a foundational base both for a functional programming language (such as Haskell) and for constructive mathematics.

In this paper we show that a typical inductive data type as that of well-founded trees has a topological counterpart in the point-free approach to topology, called formal topology.

Formal topology was introduced in [20] to develop topology in a constructive and predicative foundation such as Martin-Löf’s type theory in [16]. With respect to the usual notion of “locale” it employs a severe distinction between a set of basic opens and a collection or class (not generally a set!) of formal opens which are defined as fix-points of a closure operator on the set of basic opens.

Then, the need of building predicative and constructive examples of formal topologies, including the point-free topology of Dedekind real numbers, inspired the advent of powerful inductive methods of topological generation put forward in [3]. Since then, it was clear that some kind of well-founded set constructor was enough to formalize such a topological induction in Martin-Löf’s type theory as shown in detail in [21]. Moreover, it was also underlined that the main difficulty in generating inductive topologies reduces essentially to that of generating inductive suplattices, named *inductively generated basic covers*, because the structure of inductive frame can be easily instantiated as a special case of inductive suplattice as shown in [3] and extensively explained in [1, 2].

¹ Email: maietti@math.unipd.it

² Email: sabelli@math.unipd.it

Recently, in [11], the Curry-Howard representation of intuitionistic connectives and quantifiers as types has been extended by giving a proof-relevant presentation of inductively generated basic covers within a two-level extension of the Minimalist Foundation [9]. Moreover, combining Th.4.9 of [11] with Th.5.3 of [7] it follows that a version of Martin-Löf’s type theory with W-types has the same proof-theoretic strength as the one with inductively generated basic covers. This led to the following question: *can we establish directly in some version of Martin-Löf’s type theory an equivalence between W-types and proof-relevant inductively generated basic covers?*

Inspired by the results in [17] and [8], in this paper, we show that over intensional Martin-Löf’s type theory, W-types and proof-relevant inductive basic covers can encode one another provided that:

- (i) function extensionality holds;

or, alternatively, that

- (ii) definitional η -equalities for Π -types, Σ -types and the singleton type \mathbf{N}_1 hold.

Notice that the first hypothesis is satisfied by Homotopy Type Theory (see [19]), and the required η -equalities in the second set of hypotheses are usually implemented by default in the Agda proof-assistant.

The result we prove will actually involve two other type constructors, namely, dependent W-types, a generalisation of W-types introduced in [18] and *well-founded predicates*, a new type constructor which is a logical version of dependent W-types for predicates. Finally, we will discuss the meaning of those type constructors and their relationship in the Minimalist Foundation.

Structure of the paper

The paper is structured as follows: in Section 2, we present the two type theories we will work in and the formal notions of propositional and definitional encoding for the two type constructors; in Section 3, we recall some basic notions of formal topology, leading to the definition of inductive basic cover; in Section 4, we recall W-types and dependent W-types, showing that they are mutually encodable in some type theories; in Section 5, we introduce the type constructor of well-founded predicates, showing in what type theories it is mutually encodable with respect to both W-types and inductive basic covers.

Contributions and related works

It is well known that W-types can encode inductive datatypes. This was first proved in [4], and recently in [8] without the use of function extensionality. The fact that W-types can encode also inductive families, represented by *dependent* W-types, was first proved in extensional type theory in [17] and then in a categorical setting in [6]; in Homotopy Type Theory, a similar result concerning a slightly different generalization of W-types, namely *indexed* W-types, has also been proved (see [15]).

In [3] and [21], it was observed that the inductive generation of basic covers can be seen as a particular instance of the dependent W-type constructor.

Our main contribution is to formally show that, under some mild extensional hypotheses, extending an intensional dependent type theory with inductive generation of basic covers is equivalent to extend it with well-founded trees. To make this statement precise, we introduce the notion of *encoding* between two type constructors. Finally, we discuss what equivalences survive in the Minimalist Foundation.

The results of this paper concerning the encoding of constructors in extensions of Martin-Löf’s type theory have been checked in Agda; the formalization is available at [the second author’s GitHub Page](#).

2 Preliminaries

In this paper we work with two different intensional type theories. Namely, an intensional Martin-Löf’s type theory and the intensional level of the Minimalist Foundation. We briefly recall here both.

Martin-Löf’s type theory

We consider a version of intensional Martin-Löf’s type theory \mathbf{MLTT}_0 with the following type constructors: the empty type \mathbf{N}_0 , the unit type \mathbf{N}_1 , dependent sums Σ , dependent products Π , identity types

Id , disjoint sums $+$, and a universe of small types \mathbf{U}_0 à la Russell closed under all the above type constructors. Inductive type constructors are defined as to allow elimination toward all (small and large) types; this will be true also for the inductive types introduced in the subsequent sections; in particular, this feature is due in order to recursively define predicates on a type. The intensionality of the theory means that judgmental equality is not reflected by propositional equality; moreover, since we perform a fine-grained analysis on the use of equality, we do not assume either η -equalities or function extensionality in our base theory; instead they will appear as additional hypothesis in statements.

Minimalist Foundation

The Minimalist Foundation is a two-level foundation conceived in [13] and finalized in [9] equipped with an intensional level \mathbf{mTT} and an extensional level \mathbf{emTT} and an interpretation of the latter in a (quotient model) of the first. Both levels are fomulated as dependent type theory à la Martin-Löf. Here, we mainly work within the intensional level \mathbf{mTT} , in which there are four kinds of types: small propositions, propositions, sets and collections (denoted respectively \mathbf{prop}_s , \mathbf{prop} , \mathbf{set} and \mathbf{col}). Small propositions are both propositions and sets, and every type is a collection, as depicted in the following square of inclusions:

$$\begin{array}{ccc}
 \mathbf{prop} & \hookrightarrow & \mathbf{col} \\
 \uparrow & & \uparrow \\
 \mathbf{prop}_s & \hookrightarrow & \mathbf{set}
 \end{array}$$

This allows one to differentiate on the one hand logical and mathematical entities, and on the other different degrees of complexity (corresponding to the usual small/large type distinction in a Martin-Löf's type theory with a universe). Small propositions include the falsum constant \perp and propositional equalities $\text{Id}(A, a, b)$ of two terms a and b in the same set A , and are closed under connectives \wedge , \vee , \Rightarrow and quantification \exists , \forall over sets; propositions include all small propositions, all propositional equalities $\text{Id}(A, a, b)$ (even when A is not a set), and are closed under all connectives and quantifiers (again, without any restriction on the domain of quantification); sets include the empty set \mathbf{N}_0 , the singleton set \mathbf{N}_1 , all small propositions (identified with the sets of their proofs) and are closed under dependent sum Σ , dependent product Π , disjoint sum $+$ and the list constructor List . Finally, each set and each proposition is a collection; moreover, collections include a universe à la Russell of small propositions \mathbf{prop}_s and are closed under dependent sum. A crucial characteristic of the Minimalist Foundation is the fact that elimination rules of propositional constructors act only for propositions; for this reason the axioms of choice or of unique choice (and their rules) are not a theorem as it is in Martin-Löf type theory, since in general one cannot produce witnesses for existential statements (see [10]).

Notation

In both theories we adopt the following notational conventions:

- we use the notation of sequents to express type-judgments instead of the original notation of natural deduction in [9, 16];
- since we work in an intensional setting, we interpret the type $\mathcal{P}(A)$ of subsets of a small type A as a setoid whose carrier is the type of predicates over A following [9], namely, the function space from A to the universe of small propositions, rendered as the large type $A \rightarrow \mathbf{U}_0$ in Martin-Löf's type theory, and as the collection $A \rightarrow \mathbf{prop}_s$ in the Minimalist Foundation;
- in a type-theoretical framework we need to be careful in distinguishing between typehood judgment, denoted here with the semicolon notation $a : A$, and the (propositional) relation of membership, denoted with the usual set-theoretical membership symbol $a \in V$, when $a : A$ and $V : \mathcal{P}(A)$; the same notation will be used when working with the intensional representation of subsets mentioned in the previous point;
- when writing inference rules, the piece of context common to all the judgments appearing in an inference rule is omitted; moreover, when we give the rules of a type constructor, we interpret the formation rule's premises as parameters of the constructor; we then take them for granted in the premises of the other constructor's rules;

- we reserve the symbol \rightarrow (*resp.* \times) as a shorthand for a non-dependent function space (*resp.* for non-dependent product spaces); moreover, when working in the Minimalist Foundation, we denote the implication connective with the arrow symbol \Rightarrow ;
- when writing lambda abstractions, we omit to write the type of the abstraction;
- we write $a =_A b$ – or just $a = b$ when the type A can be easily inferred from the context – as a shorthand for $\text{ld}(A, a, b)$;
- we write $f(a)$ as a shorthand for $\text{Ap}(f, a)$ (*resp.* $\text{Ap}_\forall(f, a)$, $\text{Ap}_\Rightarrow(f, a)$) when $a : A$ and $f : (\Pi x : A)B(x)$ (*resp.* $f : (\forall x : A)B(x)$, $f : A \Rightarrow B$).

Encodings

In the contexts of extensions of Martin-Löf’s type theory and of the intensional level of the Minimalist Foundation, we define what it means for a type constructor to *encode* another type constructor.

Definition 2.1 Let \mathbf{T} be an extension of either \mathbf{mTT} , \mathbf{emTT} or \mathbf{MLTT}_0 , and let \mathbf{C} and \mathbf{D} be two type constructors of \mathbf{T} , intended as their corresponding sets of rules. We say that \mathbf{C} *definitionally encodes* \mathbf{D} in \mathbf{T} if each new symbol appearing in \mathbf{D} can be interpreted in $\mathbf{T} + \mathbf{C}$ in such a way that all the rules of \mathbf{D} are valid under this interpretation. If \mathbf{T} is an extension of \mathbf{MLTT}_0 , we say that \mathbf{C} *propositionally encodes* \mathbf{D} in \mathbf{T} if for each A such that $\mathbf{T} + \mathbf{D}$ derives A type, there exist B and p such that $\mathbf{T} + \mathbf{C}$ derives B type and $\mathbf{T} + \mathbf{C} + \mathbf{D}$ derives $p : A \cong B$, where $A \cong B$ means

$$(\Sigma f : A \rightarrow B)(\Sigma g : B \rightarrow A)((\Pi x : A)(g(f(x)) =_A x) \times (\Pi x : B)(f(g(x)) =_B x))$$

Finally, if it happens that in \mathbf{T} both \mathbf{C} definitionally (*resp.* propositionally) encodes \mathbf{D} , and \mathbf{D} definitionally (*resp.* propositionally) encodes \mathbf{C} , we say that \mathbf{C} and \mathbf{D} are *definitionally* (*resp.* propositionally) *mutually encodable*.

As already mentioned, we will consider extending a theory with the following axioms:

(i) function extensionality

$$\frac{f : (\Pi x : A)B(x) \quad g : (\Pi x : A)B(x) \quad p : (\Pi x : A)(f(x) =_{B(x)} g(x))}{\text{funext}(f, g, p) : f =_{(\Pi x : A)B(x)} g}$$

(ii) η -equality for Π -types

$$\frac{f : (\Pi x : A)B(x)}{f = \lambda x. f(x) : (\Pi x : A)B(x)}$$

(iii) η -equality for Σ -types

$$\frac{z : (\Sigma x : A)B(x)}{z = \langle \text{pr}_1(z), \text{pr}_2(z) \rangle : (\Sigma x : A)B(x)}$$

(iv) η -equality for \mathbf{N}_1

$$\frac{z : \mathbf{N}_1}{z = \star : \mathbf{N}_1}$$

3 Inductive basic covers

The name “Formal Topology” refers both to the study of topology in a constructive and predicative setting and to its main object of investigation: a point-free notion of topology whose definition, contrary to the classical one of topological space, avoids impredicative uses of the powerset. The core component of a formal topology is its underlying *basic cover*, a predicative presentation of the topology’s suplattice of open sets. We recall here its definition as originally appeared in [1].

Definition 3.1 A *basic cover relation* on a set A is a binary relation $a \triangleleft V$ between elements $a : A$ and subsets $V : \mathcal{P}(A)$, such that the following two properties hold:

- (i) (*reflexivity*) if $a \in V$, then $a \triangleleft V$;
- (ii) (*transitivity*) if $a \triangleleft U$, and $u \triangleleft V$ for each $u \in U$, then $a \triangleleft V$.

In this paper, we are specifically interested in a subclass of basic covers; namely the *inductively generated* ones. The inductive generation of basic covers was devised by the authors in [3] to have a convenient way for constructing formal topologies. Moreover, their method enjoys some desirable properties, such as the possibility of forming the product topology of two inductively generated formal topologies, which in general does not appear definable predicatively. We recall the notion of inductively generated basic cover, starting with the definition of axiom set.

Definition 3.2 An *axiom set* consists of:

- (i) a set A ;
- (ii) an A -indexed family of sets $a : A \vdash I(a)$ set ;
- (iii) for each $a : A$, an $I(a)$ -indexed family of A 's subsets $a : A, i : I(a) \vdash C(a, i) : \mathcal{P}(A)$.

The subsets family C in the definition above is to be understood as a collection of axioms (hence the name axiom set) of the form $a \triangleleft C(a, i)$ for each $a : A$ and each $i : I(a)$. Given an axiom set, the basic cover inductively generated by it is the smallest basic cover that satisfies those axioms, formally:

Definition 3.3 A basic cover *inductively generated* by an axiom set A, I, C is a basic cover \triangleleft on A such that:

- (i) $a \triangleleft C(a, i)$ holds for each $a : A$ and $i : I(a)$;
- (ii) if \triangleleft' is another basic cover such that $a \triangleleft' C(a, i)$ holds for each $a : A$ and $i : I(a)$, then $a \triangleleft V$ implies $a \triangleleft' V$ for each $a : A$ and $V : \mathcal{P}(A)$.

If a basic cover happens to be inductively generated by some axiom set, we say that it is an *inductive basic cover*.

Given an axiom set A, I, C , it is always possible to construct the basic cover inductively generated by it. It is the relation $\triangleleft_{I,C}$ obtained using the following generating rules:

- (i) (*reflexivity*) if $a \in V$, then $a \triangleleft_{I,C} V$;
- (ii) (*infinity*) if, for some $a : A$ and $i : I(a)$, it holds that $b \triangleleft_{I,C} V$ for each $b \in C(a, i)$, then $a \triangleleft_{I,C} V$.

In [11], a new type constructor was added both in the Minimalist Foundation (A) and, following the Curry-Howard interpretation, in Martin-Löf's type theory (A) for interpreting the above generating method at an intensional level; there, examples of inductively basic covers were also defined, including the inductive topology of Dedekind real numbers and the inductive topology of the Cantor or Baire space. The type constructor follows the usual scheme for inductive types and in the following sections we will investigate how it compares to W-types and their dependent versions.

4 Well-founded trees constructors

The W-type constructor (also known as the type of well-orders or well-founded trees) was present in the first versions of Martin-Löf's type theory (see [14]), where it was used to constructively represent ordinals. Its rules in Martin-Löf's type theory are listed in A. For a set A and an A -indexed family of sets B , the W-type $W_{A,B}$ is generally understood set-theoretically as the set of (possibly infinitary) well-founded trees with nodes labelled by elements of a set A and with a branching function given by B .

One of the main reasons for the importance of W-types is their ability, first observed by Dybjer in [4], to encode common inductive types such as natural numbers and lists. Our goal is to show that this is also the case for the (Curry-Howard presentation of) inductive basic covers. However, as soon as we confront the W-type constructor with that of inductive basic covers, we discover a substantial issue: the former produces just a set, while the latter produces a predicate, hence, by the propositions-as-types paradigm, a family of sets. This same limitation of the W-type constructor has been already addressed in [17], where they proposed a generalization of W-types, called *dependent W-types* (also known as *indexed W-types*, or simply as *trees*), capable of constructing *families of mutually* inductive sets. This additional expressivity

allows for example the construction of W-types in the setoid model without having to eliminate towards the universe, as shown in [5].

The rules of dependent W-types in Martin-Löf's type theory are listed in A. Also dependent W-types can be interpreted as sets of well-founded trees, but their structure is more complex. The nodes of a dependent W-type are labelled by elements of a set I as in the non-dependent case, however there is not just one branching associated to each label; instead, each label $i : I$ allows between a set $N(i)$ of possible branchings. Actually, an element $n : N(i)$ is just a label for the branching, the branching itself being given by a sets family $Br(i, n)$. Lastly, the trees of a dependent W-type satisfy a further property: the roots of each of its directed subtrees must be labelled according to an *arity* function $ar(i, n) : Br(i, n) \rightarrow I$. We write a dependent W-type in symbol as $DW_{Br, ar}$, omitting the parameters I and N for readability.

Of course, W-types are a particular instance of dependent W-types. This can be formally seen by setting the parameter I to be the unit type \mathbf{N}_1 . Actually, and perhaps surprisingly, also the converse is true, namely dependent W-types can be reduced to non-dependent ones. In the following proposition we spell out our result, which is essentially a recasting in an intensional setting of the ideas contained in [17] and [6].

Proposition 4.1 (i) *W-types propositionally encode dependent W-types in \mathbf{MLTT}_0 extended with function extensionality;*

(ii) *W-types definitionally encode dependent W-types in \mathbf{MLTT}_0 extended with η -equalities for Π -types and Σ -types.*

Proof. Assume to have the parameters

$$\begin{aligned} I &: \mathbf{U}_0 \\ i : I &\vdash N(i) : \mathbf{U}_0 \\ i : I, n : N(i) &\vdash Br(i, n) : \mathbf{U}_0 \\ i : I, n : N(i) &\vdash ar(i, n) : Br(i, n) \rightarrow I \end{aligned}$$

as in the premises of the formation rule of dependent W-types. Firstly, we construct a W-type \mathbf{Free} of well-founded trees whose nodes are labelled by dependent pairs in $(\Sigma i : I)N(i)$ and whose branching function is given by the family Br applied to the two projections. Formally, we are constructing the set

$$\mathbf{Free} := \mathbf{W}_{(\Sigma i : I)N(i), Br(\text{pr}_1(z), \text{pr}_2(z))}$$

This means that \mathbf{Free} trees' nodes contain information both on how the nodes of the dependent W-type trees we are trying to simulate are labelled and on the branching options chosen for those nodes; all while respecting the same branching function.

Secondly, we impose that each node's label is in accordance with the arity parameter ar by thinning out the set of \mathbf{Free} trees with a I -indexed family of predicates $i : I \vdash \mathbf{Legal}(i) : \mathbf{Free} \rightarrow \mathbf{U}_0$ which assert, for a \mathbf{Free} tree, that its root's label has i as its first component, and that each other node's first component is given by the arity function applied to its parent's labels and branch. The predicate is formally defined by recursion for any $i : I, j : I, n : N(j)$ and $f : Br(j, n) \rightarrow \mathbf{Free}$ in the following way:

$$\mathbf{Legal}(i, \text{sup}(\langle j, n \rangle, f)) := (\Pi b : Br(j, n)) \mathbf{Legal}(ar(j, n, b), f(b)) \times (i =_I j)$$

Note that, being i fixed, the identity type $i =_I j$ has a unique proof propositionally, and hence it is contractible type according to [19].

Then, our candidate for encoding the dependent W-type is the type family

$$i : I \vdash DW'(i) := (\Sigma w : \mathbf{Free}) \mathbf{Legal}(i, w)$$

Now suppose to have $i : I, n : N(i)$ and $f : (\Pi b : Br(i, n)) DW'(ar(i, n, b))$, then we can straightforwardly define a constructor term of $DW'(i)$ in the following way:

$$\mathbf{dsup}'(i, n, f) := \langle \text{sup}(\langle i, n \rangle, \lambda b. \text{pr}_1(f(b))), \langle \lambda b. \text{pr}_2(f(b)), \text{id}(i) \rangle \rangle$$

For (i), we can now define by recursion a pair of functions $g_i : DW_{Br,ar}(i) \rightarrow DW'(i)$ and $g_i^{-1} : DW'(i) \rightarrow DW_{Br,ar}(i)$ for each $i : I$ in the following way

$$\begin{aligned} g_i(\mathbf{dsup}(i, n, f)) &::= \mathbf{dsup}'(i, n, \lambda b. g_i(f(b))) \\ g_i^{-1}(\langle \mathbf{sup}(\langle i, n \rangle, f), \langle l, \mathbf{id}(i) \rangle \rangle) &::= \mathbf{dsup}(i, n, \lambda b. g_i^{-1}(\langle f(b), l(b) \rangle)) \end{aligned}$$

Using function extensionality, we can then check by induction that they are reciprocally inverses.

For (ii), we have already derived the formation and introduction rules of dependent W-types with DW' and \mathbf{dsup}' . Elimination and computation rules are a bit more involved, although they are not conceptually harder. Suppose to have, as in the premises of the elimination rule, a type family $i : I, w : DW'(i) \vdash M(i, w)$ and a dependent term

$$\begin{aligned} i : I, \\ n : N(i), \\ f : (\Pi b : Br(i, n)) DW'(ar(i, n, b)), \\ h : (\Pi b : Br(i, n)) M(ar(i, n, b), f(b)) \\ \vdash d(i, n, f, h) : M(i, \mathbf{dsup}'(i, n, f)) \end{aligned}$$

For the elimination rule we want to define a dependent term

$$i : I, w : DW'(i) \vdash \mathbf{El}'_{DW}(i, w, d) : M(i, w)$$

satisfying the following definitional equality to validate the computational rule

$$\begin{aligned} i : I \\ n : N(i) \\ f : (\Pi b : Br(i, n)) DW'(ar(i, n, b)) \\ \vdash \mathbf{El}'_{DW}(i, \mathbf{dsup}'(i, n, f), d) = d(i, n, f, \lambda b. \mathbf{El}'_{DW}(ar(i, n, b), f(b), d)) \end{aligned}$$

The idea is to define the term \mathbf{El}'_{DW} by recursion by mimicking the above requirement – so that the task of checking the computation rule will turn out to be trivial. The definition explicitly reads

$$\mathbf{El}'_{DW}(i, \langle \mathbf{sup}(\langle i, n \rangle, f), l, \mathbf{id}(i) \rangle, d) ::= d(i, n, \lambda b. \langle f(b), l(b) \rangle, \lambda b. \mathbf{El}'_{DW}(ar(i, n, b), \langle f(b), l(b) \rangle))$$

Above, we used the recursion principles of Σ -types, W-types and identity types, all at once. The long-but-routine calculations lie in checking that the given definition is well-typed by formulating it only with eliminator terms. In particular, it is in this step of defining \mathbf{El}'_{DW} by recursion that η -equalities are needed to ensure that the calculations go through. We leave them to the assiduous reader or to the proof-checker. \square

Taking advantage of the previous result, we will show the equivalence between W-types and inductive basic covers by proving, on the one hand, that dependent W-types can encode them, and, on the other, that (non-dependent) W-types can be encoded by them. However, we will not prove these two facts directly; instead, in the next section, we will introduce a new type constructor, called *well-founded predicate*, to use as a bridge between (dependent) W-types and inductive basic covers. This intermediate step would not be strictly necessary for Martin-Löf's type theory; nevertheless, aside from providing a clearer proof, its introduction is vital when we wish to keep apart the notions of set and proposition as it is done in the Minimalist Foundation.

We close this section with a remark on the behaviour of W-types in the Minimalist Foundation which provides another, more technical reason why it is not convenient to work with them in such a framework.

Remark 4.2 We know that in Martin-Löf's type theory, without a universe of sets, it is not possible to construct a family $x : A \vdash B(x)$ set for which there exist $a, a' : A$ such that $B(a)$ is inhabited and $B(a')$

is (isomorphic to) the empty type. In the Minimalist Foundation, in which there is only a universe of (small) propositions, the same phenomenon occurs, with an additional subtlety: in fact, we can construct a non-always-inhabited, non-always-empty family of sets, e.g. $b : \mathbf{N}_1 + \mathbf{N}_1 \vdash \text{El}_+(b, \perp, \top)$; however, no set of such a family seems to be provable isomorphic to the empty set \mathbf{N}_0 , the main reason being that the falsum constant \perp cannot eliminate towards sets. Consequently, in the Minimalist Foundation we cannot start the construction of a canonical element for a W-type (aside for those with a constantly-null branching function) because, according to the W-type introduction rule, that would require to construct a function towards the W-type itself, which is again a set.

Nonetheless, a suitable extension of the Minimalist Foundation (e.g. one equipped with a universe of sets) could make W-types (and their dependent version) actually usable; a result analogous to point (ii) of 4.1 can be then shown for the intensional level of the Minimalist Foundation.

5 Well-founded predicates

As mentioned in the previous section, the choice of representing predicates (such as inductive basic covers) through sets (such as well-founded trees) would not be consistent with the Minimalist Foundation's philosophy, which stipulates a strict distinction between propositions and sets. This brings us to introduce a new propositional constructor called *well-founded predicate* to the Minimalist Foundation that allows for the inductive definition of predicates.

This new constructor stems as a logical counterpart for predicates of the notion of dependent well-founded tree, hence the name well-founded predicate. Analogously, a well-founded predicate can be interpreted as the set of *proof* trees built using certain derivation rules defined by its parameters in the following way: I is the set on which the well-founded predicate acts; for each element $i : I$, $N(i)$ is a set of names for inference rules whose conclusions state that the well-founded predicate holds for i ; the premises of each rule $n : N(i)$ are (possibly infinite) instances of the well-founded predicate applied to some elements of I , and the predicate $R(i, n) : \mathcal{P}(I)$ tells precisely which ones. Schematically, we have the following entailments written in the language of the extensional level of the Minimalist Foundation:

$$i : I, n : N(i) \vdash (\forall j \in R(i, n)) \text{WP}_R(j) \Rightarrow \text{WP}_R(i)$$

In particular, if $R(i, n)$ is the empty subset of I , the above entailment is interpreted as an axiom stating that the predicate holds for i ; from there, the construction of a derivation can start (notice that we do not encounter the same problem for well-founded trees pointed out in 4.2, since now the well-founded predicate is a proposition towards which the falsum constant can eliminate).

The following representation property wraps up the above interpretation as a provable statement in the extensional level of the Minimalist Foundation:

$$i : I \vdash \text{WP}_R(i) \Leftrightarrow (\exists n : N(i)) (\forall j \in R(i, n)) \text{WP}_R(j)$$

It explicitly reads: “the well-founded predicate holds for i if and only if there exists $n : N(i)$ such that all the premises $R(i, n)$ of the rules are satisfied by the well-founded predicate”.

In order to interpret this extensional predicate in the intensional level in a similar way to what is done in Proposition 2.7 of [11] for the inductive cover predicate, it is enough to extend the intensional level of the Minimalist Foundation with the rules of the W-predicate constructors in the appendix A.

Of course, the motivating example for introducing well-founded predicates is the inductive generation of basic covers. To formally see this, suppose to have an axiom set A, I, C and a predicate $V : A \rightarrow \mathbf{prop}_s$; we can express the predicate $a \triangleleft_{I, C} V$ as a well-founded predicate over A constructed using the following parameters:

$$\begin{aligned} N(a) &::= V(a) + I(a) \\ R(a, n) &::= \begin{cases} \emptyset & \text{if } n : V(a) \\ C(a, i) & \text{if } n : I(a) \end{cases} \end{aligned}$$

If we also have at our disposal the basic inductive cover constructor, we can then form and prove in the theory the statement $a : A \vdash a \triangleleft_{I,C} V \Leftrightarrow \text{WP}_R(a)$ asserting the equivalence between the two predicates.

Actually, the basic inductive covers are a complete example of a well-founded predicate, in the sense that their scope encompasses all the possibilities of the constructor. This is perhaps not surprising since, although interpreted differently, the rules of the former resemble very closely the ones of the latter, the only difference being the additional canonical elements introduced by reflexivity. To formally see this, suppose to have I, N, R parameters for the well-founded predicates constructor; the statement $i : I \vdash \text{WP}_R(i) \Leftrightarrow i \triangleleft_{N,R} \emptyset$ is then provable in the theory.

Of course, as for inductive basic covers, well-founded predicates can be defined also in Martin-Löf's type theory by defining the elimination rule to act towards any dependent type (we spell out the rules in [A](#)), according to the propositions-as-types correspondence. Then, the previous results on the relationship between inductive basic covers and well-founded predicates based on equiprovability upgrade to the following statement.

Proposition 5.1 (i) *well-founded predicates are propositionally mutually encodable with inductive basic covers in \mathbf{MLTT}_0 extended with function extensionality;*
 (ii) *well-founded predicates are definitionally mutually encodable inductive basic covers in \mathbf{MLTT}_0 and \mathbf{mTT} , both extended with η -equalities for Π -types and Σ -types, and also in \mathbf{emTT} .*

Proof. Showing that inductive basic covers encode well-founded predicates in \mathbf{MLTT}_0 is a straightforward adaptation of the construction presented above in the case of the Minimalist Foundation.

The reverse is more involved, since it requires employing the technique presented in [\[8\]](#) to correctly encode the reflexivity case of inductive basic covers. We show it only in the case of \mathbf{MLTT}_0 , the case for \mathbf{mTT} being entirely analogous.

We need to refine the construction with a predicate family

$a : A \vdash \text{Canonical}(a) : \text{WP}_R(a) \rightarrow \mathbf{U}_0$ defined by recursion in the following way (we leave implicit the dependence on a of the predicate):

$$\text{Canonical}(\text{ind}(a, n, f)) := \begin{cases} f =_{(\Pi j : I)(\mathbf{N}_0 \rightarrow \text{WP}_R(j))} \lambda j. \lambda z. \text{El}_{\mathbf{N}_0}(z) & \text{if } n : V(a) \\ (\Pi b : A)(\Pi r : R(a, n, b)) \text{Canonical}(f(b, r)) & \text{if } n : I(a) \end{cases}$$

Reasoning analogously as in [4.1](#), it is easy to check that the type family

$$a : A \vdash (\Sigma w : \text{WP}_R(a)) \text{Canonical}(w)$$

is isomorphic to the inductive basic cover $a \triangleleft_{I,C} V$ (assuming function extensionality); or that it satisfies the rules of the inductive basic covers constructor (assuming η -equalities). \square

We mentioned how well-founded predicates were added to the Minimalist Foundation as a natural propositional counterpart of dependent W-types. In Martin-Löf's type theory we can show that the two constructors are indeed equivalent; to achieve this, we combine [4.1](#) with the following result.

Proposition 5.2 (i) *Dependent W-types propositionally encode well-founded predicates in \mathbf{MLTT}_0 extended with function extensionality;*
 (ii) *Dependent W-types definitionally encode well-founded predicates in \mathbf{MLTT}_0 extended with η -equality for Σ -types and Π -types;*
 (iii) *Well-founded predicates propositionally encode dependent W-types in \mathbf{MLTT}_0 extended with function extensionality;*
 (iv) *Well-founded predicates definitionally encode dependent W-types in \mathbf{MLTT}_0 extended with η -equality for Σ -types and the unit type \mathbf{N}_1 .*

Proof.

To show that dependent W-types encode well-founded predicates, assume to have parameters I, N, R and recall that

$$R : (\Pi i : I)(N(i) \rightarrow (I \rightarrow \mathbf{U}_0))$$

To construct the desired dependent W-type, we take for I and N the parameters with the same names; while, for each $i : I$ and $n : N(i)$, we define

$$\begin{aligned} Br(i, n) &::= (\Sigma j : I)R(i, n, j) \\ ar(i, n) &::= \text{pr}_1 : Br(i, n) \rightarrow I \end{aligned}$$

Without any further intricacies, the type family $i : I \vdash DW_{(\Sigma j : I)R(i, n, j), \text{pr}_1}$ can be proved to satisfy, under the chosen hypotheses, the corresponding statement as in propositions 4.1 and 5.1.

To show that well-founded predicates encode W-types, assume to have the W-type parameters $A : \mathbf{U}_0$ and $B : A \rightarrow \mathbf{U}_0$. Firstly, we construct a well-founded predicate choosing for I the unit type \mathbf{N}_1 , for N the constant type family at A , and for R the type family B depending only on the A 's indexing. We then instantiate the type family $x : \mathbf{N}_1 \vdash WP_R(x) : \mathbf{U}_0$ obtained in this way at the canonical element of the unit type \star , obtaining a type $W' ::= WP_R(\star) : \mathbf{U}_0$; it is straightforward to check that this type encodes the W-type constructor. \square

6 Conclusions

We can combine propositions 4.1, 5.2 and 5.1 to show, under the propositions-as-type paradigm, the essential equivalence of all the type constructors previously considered:

Theorem 6.1 *The following type constructors are all propositionally mutually encodable in \mathbf{MLTT}_0 extended with function extensionality, and definitionally mutually encodable in \mathbf{MLTT}_0 extended with η -equalities for Π -types, Σ -types and \mathbf{N}_1 :*

- (i) *W-types;*
- (ii) *dependent W-types;*
- (iii) *well-founded predicates;*
- (iv) *inductive basic covers.*

In particular, we have the following corollaries:

Corollary 6.2 (i) *W-types and inductive basic covers are propositionally mutually encodable in Homotopy Type Theory;*
 (ii) *W-types and inductive basic covers are definitionally mutually encodable in the Agda implementation of intensional Martin-Löf's type theory.*

Finally, combining 4.2 and 5.1, we get a corresponding result for the intensional level of the Minimalist Foundation extended with η -equalities for Σ - and Π -types, and in \mathbf{emTT} , which already satisfies η -equalities. Clearly, the equivalence is no longer extended to all four type constructors since propositions and types are no longer identified:

Theorem 6.3 *In \mathbf{mTT} extended with η -equalities for Σ - and Π -types, and in \mathbf{emTT} the followings hold:*

- (i) *W-types and dependent W-types are definitionally mutually encodable;*
- (ii) *basic inductive covers and well-founded predicates are definitionally mutually encodable.*

Conclusions and future work

We have shown that in Martin-Löf's type theory, an axiomatic treatment of inductive topology is not less powerful than the addition of W-types. In the Minimalist Foundation, this is true when W-types are replaced by well-founded predicates, given that the identification of all types as propositions is no longer valid. In future works, we hope to extend the comparison to the topological positivity relation defined in [12] with suitable coinductive predicates.

References

- [1] Battilotti, G. and G. Sambin, *Pretopologies and a uniform presentation of sup-lattices, quantales and frames*, Ann. Pure Appl. Logic **137** (2006), pp. 30–61.
URL <https://doi.org/10.1016/j.apal.2005.05.017>
- [2] Ciraulo, F., M. E. Maietti and G. Sambin, *Convergence in formal topology: a unifying notion*, J. Log. Anal. **5** (2013), pp. Paper 2, 45.
URL <https://doi.org/10.4115/jla.2013.5.2>
- [3] Coquand, T., G. Sambin, J. Smith and S. Valentini, *Inductively generated formal topologies*, Ann. Pure Appl. Logic **124** (2003), pp. 71–106.
URL [https://doi.org/10.1016/S0168-0072\(03\)00052-6](https://doi.org/10.1016/S0168-0072(03)00052-6)
- [4] Dybjer, P., *Representing inductively defined sets by wellorderings in Martin-Löf’s type theory*, Theoret. Comput. Sci. **176** (1997), pp. 329–335.
URL [https://doi.org/10.1016/S0304-3975\(96\)00145-4](https://doi.org/10.1016/S0304-3975(96)00145-4)
- [5] Emmenegger, J., *W-types in setoids*, Log. Methods Comput. Sci. **17** (2021), pp. Paper No. 28, 33.
URL [https://doi.org/10.46298/lmcs-17\(3:28\)2021](https://doi.org/10.46298/lmcs-17(3:28)2021)
- [6] Gambino, N. and M. Hyland, *Wellfounded trees and dependent polynomial functors*, in: *Types for proofs and programs*, Lecture Notes in Comput. Sci. **3085**, Springer, Berlin, 2004 pp. 210–225.
URL https://doi.org/10.1007/978-3-540-24849-1_14
- [7] Griffor, E. and M. Rathjen, *The strength of some Martin-Löf type theories*, Archive for Mathematical Logic **33** (1994), pp. 347–385.
- [8] Hugunin, J., *Why Not W?*, in: U. de’Liguoro, S. Berardi and T. Altenkirch, editors, *26th International Conference on Types for Proofs and Programs (TYPES 2020)*, Leibniz International Proceedings in Informatics (LIPIcs) **188** (2021), pp. 8:1–8:9.
URL <https://drops.dagstuhl.de/opus/volltexte/2021/13887>
- [9] Maietti, M. E., *A minimalist two-level foundation for constructive mathematics*, Ann. Pure Appl. Logic **160** (2009), pp. 319–354.
URL <https://doi.org/10.1016/j.apal.2009.01.006>
- [10] Maietti, M. E., *On choice rules in dependent type theory*, in: *Theory and applications of models of computation*, Lecture Notes in Comput. Sci. **10185**, Springer, Cham, 2017 pp. 12–23.
URL https://doi.org/10.1007/978-3-319-55911-7_2
- [11] Maietti, M. E., S. Maschio and M. Rathjen, *A realizability semantics for inductive formal topologies, Church’s thesis and axiom of choice*, Log. Methods Comput. Sci. **17** (2021), pp. Paper No. 21, 21.
URL [https://doi.org/10.23638/LMCS-17\(2:21\)2021](https://doi.org/10.23638/LMCS-17(2:21)2021)
- [12] Maietti, M. E., S. Maschio and M. Rathjen, *Inductive and coinductive topological generation with Church’s thesis and the axiom of choice*, Log. Methods Comput. Sci. **18** (2022), pp. Paper No. 5, 28.
URL [https://doi.org/10.46298/lmcs-18\(4:5\)2022](https://doi.org/10.46298/lmcs-18(4:5)2022)
- [13] Maietti, M. E. and G. Sambin, *Toward a minimalist foundation for constructive mathematics*, in: *From sets and types to topology and analysis*, Oxford Logic Guides **48**, Oxford Univ. Press, Oxford, 2005 pp. 91–114.
URL <https://doi.org/10.1093/acprof:oso/9780198566519.003.0006>
- [14] Martin-Löf, P., “Intuitionistic Type Theory, notes by G. Sambin of a series of lectures given in Padua, June 1980,” Bibliopolis, Naples, 1984.
- [15] nLab authors, *inductive family*, <https://ncatlab.org/nlab/show/inductive+family> (2023), [Revision 25](#).
- [16] Nordström, B., K. Petersson and J. Smith, “Programming in Martin Löf’s Type Theory.” Clarendon Press, Oxford, 1990.
- [17] Petersson, K. and D. Synek, *A set constructor for inductive sets in Martin-Löf’s type theory*, in: *Category theory and computer science (Manchester, 1989)*, Lecture Notes in Comput. Sci. **389**, Springer, Berlin, 1989 pp. 128–140.
URL <https://doi.org/10.1007/BFb0018349>
- [18] Petersson, K. and D. Synek, *A set constructor for inductive sets in martin-löf’s type theory*, in: *Category Theory and Computer Science, Manchester, UK, September 5-8, 1989, Proceedings*, 1989, pp. 128–140.
URL <https://doi.org/10.1007/BFb0018349>
- [19] Program, T. U. F., “Homotopy type theory—univalent foundations of mathematics,” The Univalent Foundations Program, Princeton, NJ; Institute for Advanced Study (IAS), Princeton, NJ, 2013, xiv+589 pp.

[20] Sambin, G., *Intuitionistic formal spaces - a first communication*, Mathematical logic and its applications (1987), pp. 187–204.

[21] Valentini, S., *Constructive characterizations of bar subsets*, Ann. Pure Appl. Logic **145** (2007), pp. 368–378.
URL <https://doi.org/10.1016/j.apal.2006.10.003>

A Rules of type constructors

Rules for well-founded trees in Martin-Löf's type theory

$$\frac{A : \mathbf{U}_0 \quad x : A \vdash B(x) : \mathbf{U}_0}{W_{A,B} : \mathbf{U}_0} \text{F-W}$$

$$\frac{a : A \quad f : B(a) \rightarrow W_{A,B}}{\text{sup}(a, f) : W_{A,B}} \text{I-W}$$

$$\frac{\begin{array}{l} w : W_{A,B} \vdash M(w) \text{ set} \quad w : W_{A,B} \\ a : A, f : B(a) \rightarrow W_{A,B}, h : (\Pi b : B(a))M(f(b)) \vdash d(a, f, h) : M(\text{sup}(a, f)) \end{array}}{\text{El}_W(w, d) : M(w)} \text{E-W}$$

$$\frac{\begin{array}{l} w : W_{A,B} \vdash M(w) \text{ set} \quad a : A \quad f : B(a) \rightarrow W_{A,B} \\ a : A, f : B(a) \rightarrow W_{A,B}, h : (\Pi b : B(a))M(f(b)) \vdash d(a, f, h) : M(\text{sup}(a, f)) \end{array}}{\text{El}_W(\text{sup}(a, f), d) = d(a, f, \lambda b. \text{El}_W(f(b), d)) : M(\text{sup}(a, f))} \text{C-W}$$

Rules for dependent well-founded trees in Martin-Löf's type theory

$$\frac{\begin{array}{l} I : \mathbf{U}_0 \\ i : I \vdash N(i) : \mathbf{U}_0 \\ i : I, n : N(i) \vdash Br(i, n) : \mathbf{U}_0 \\ i : I, n : N(i) \vdash ar(i, n) : Br(i, n) \rightarrow I \end{array}}{\text{DW}_{Br,ar} : I \rightarrow \mathbf{U}_0} \text{F-DW}$$

$$\frac{i : I \quad n : N(i) \quad f : (\Pi b : Br(i, n))\text{DW}_{Br,ar}(ar(i, n, b))}{\text{dsup}(i, n, f) : \text{DW}_{Br,ar}(i)} \text{I-DW}$$

$$\frac{\begin{array}{l} i : I, w : \text{DW}_{Br,ar}(i) \vdash M(i, w) : \mathbf{U}_0 \\ i : I \quad w : \text{DW}_{Br,ar}(i) \\ i : I, \\ n : N(i), \\ f : (\Pi b : Br(i, n))\text{DW}_{Br,ar}(ar(i, n, b)), \\ h : (\Pi b : Br(i, n))M(ar(i, n, b), f(b)) \\ \vdash d(i, n, f, h) : M(i, \text{dsup}(i, n, f)) \end{array}}{\text{El}_{\text{DW}}(i, w, d) : M(i, w)} \text{E-DW}$$

$$\begin{array}{c}
 i : I, w : \text{DW}_{Br,ar}(i) \vdash M(i, w) : \mathbf{U}_0 \\
 i : I \quad n : N(i) \quad f : (\prod_b : Br(i, n)) \text{DW}_{Br,ar}(ar(i, n, b)) \\
 i : I, \\
 n : N(i), \\
 f : (\prod_b : Br(i, n)) \text{DW}_{Br,ar}(ar(i, n, b)), \\
 h : (\prod_b : Br(i, n)) M(ar(i, n, b), f(b)) \\
 \vdash d(i, n, f, h) : M(i, \text{dsup}(i, n, f)) \\
 \hline
 \text{El}_{\text{DW}}(i, \text{dsup}(i, n, f), d) = d(i, n, f, \lambda b. \text{El}_{\text{DW}}(ar(i, n, b), f(b), d)) : M(i, \text{dsup}(i, n, f)) \quad \text{C-DW}
 \end{array}$$

Rules for inductive basic covers in mTT

$$\begin{array}{c}
 A \text{ set} \\
 a : A \vdash I(a) \text{ set} \\
 a : A, i : I(a) \vdash C(a, i) : A \rightarrow \text{prop}_s \\
 V : A \rightarrow \text{prop}_s \\
 \hline
 a : A \vdash a \triangleleft_{I,C} V \text{ prop}_s \quad \text{F} - \triangleleft \\
 \\
 \frac{a : A \quad r : V(a)}{\text{rf}(a, r) : a \triangleleft_{I,C} V} \text{Irf} - \triangleleft \\
 \\
 \frac{a : A \quad i : I(a) \quad r : (\forall b : A)(C(a, i, b) \Rightarrow b \triangleleft_{I,C} V)}{\text{tr}(a, i, r) : a \triangleleft_{I,C} V} \text{Itr} - \triangleleft \\
 \\
 \frac{a : A \vdash P(a) \text{ prop} \quad a : A \quad p : a \triangleleft_{I,C} V \\
 a : A, r : V(a) \vdash q_1(a, r) : P(a) \\
 a : A, i : I(a), s : (\forall b : A)(C(a, i, b) \Rightarrow P(b)) \vdash q_2(a, i, s) : P(a)}{\text{El}_{\triangleleft}(p, q_1, q_2) : P(a)} \text{E} - \triangleleft \\
 \\
 \frac{a : A \vdash P(a) \text{ prop} \quad a : A \quad r : V(a) \\
 a : A, r : V(a) \vdash q_1(a, r) : P(a) \\
 a : A, i : I(a), s : (\forall b : A)(C(a, i, b) \Rightarrow P(b)) \vdash q_2(a, i, s) : P(a)}{\text{El}_{\triangleleft}(\text{rf}(a, r), q_1, q_2) = q_1(a, r) : P(a)} \text{Crf} - \triangleleft \\
 \\
 \frac{a : A \vdash P(a) \text{ prop} \\
 a : A \quad i : I(a) \quad r : (\forall b : A)(C(a, i, b) \Rightarrow b \triangleleft_{I,C} V) \\
 a : A, r : V(a) \vdash q_1(a, r) : P(a) \\
 a : A, i : I(a), s : (\forall b : A)(C(a, i, b) \Rightarrow P(b)) \vdash q_2(a, i, s) : P(a)}{\text{El}_{\triangleleft}(\text{tr}(a, i, r), q_1, q_2) = q_2(a, i, \lambda \forall b. \lambda \Rightarrow t. \text{El}_{\triangleleft}(r(b, t), q_1, q_2)) : P(a)} \text{Ctr} - \triangleleft
 \end{array}$$

Rules for inductive basic covers in Martin-Löf's type theory

$$\begin{array}{c}
 A : \mathbf{U}_0 \\
 a : A \vdash I(a) : \mathbf{U}_0 \\
 a : A, i : I(a) \vdash C(a, i) : A \rightarrow \mathbf{U}_0 \\
 V : A \rightarrow \mathbf{U}_0 \\
 \hline
 a : A \vdash a \triangleleft_{I,C} V : \mathbf{U}_0 \quad \text{F} - \triangleleft
 \end{array}$$

$$\frac{a : A \quad r : V(a)}{\text{rf}(a, r) : a \triangleleft_{I,C} V} \text{I}_{\text{rf}} - \triangleleft$$

$$\frac{a : A \quad i : I(a) \quad r : (\Pi b : A)(C(a, i, b) \rightarrow b \triangleleft_{I,C} V)}{\text{tr}(a, i, r) : a \triangleleft_{I,C} V} \text{I} - \triangleleft$$

$$\frac{\begin{array}{l} a : A, p : a \triangleleft_{I,C} V \vdash M(a, p) \text{ set} \\ a : A \quad p : a \triangleleft_{I,C} V \\ a : A, r : V(a) \vdash q_1(a, r) : M(a, \text{rf}(a, r)) \\ a : A, i : I(a), \\ r : (\Pi b : A)(C(a, i, b) \rightarrow b \triangleleft_{I,C} V), \\ h : (\Pi b : A)(\Pi s : C(a, i, b))M(b, r(b, s)) \\ \vdash q_2(a, i, r, h) : M(a, \text{tr}(a, i, r)) \end{array}}{\text{El}_{\triangleleft}(p, q_1, q_2) : M(a, p)} \text{E} - \triangleleft$$

$$\frac{\begin{array}{l} a : A, p : a \triangleleft_{I,C} V \vdash M(a, p) \text{ set} \\ a : A \quad r : V(a) \\ a : A, r : V(a) \vdash q_1(a, r) : M(a, \text{rf}(a, r)) \\ a : A, i : I(a), \\ r : (\Pi b : A)(C(a, i, b) \rightarrow b \triangleleft_{I,C} V), \\ h : (\Pi b : A)(\Pi s : C(a, i, b))M(b, r(b, s)) \\ \vdash q_2(a, i, r, h) : M(a, \text{tr}(a, i, r)) \end{array}}{\text{El}_{\triangleleft}(\text{rf}(a, r), q_1, q_2) = q_1(a, r) : M(a, \text{rf}(a, r))} \text{C}_{\text{rf}} - \triangleleft$$

$$\frac{\begin{array}{l} a : A, p : a \triangleleft_{I,C} V \vdash M(a, p) \text{ set} \\ a : A \quad i : N(a) \quad r : (\Pi b : A)(C(a, i, b) \rightarrow b \triangleleft_{I,C} V) \\ a : A, r : V(a) \vdash q_1(a, r) : M(a, \text{rf}(a, r)) \\ a : A, i : I(a), \\ r : (\Pi b : A)(C(a, i, b) \rightarrow b \triangleleft_{I,C} V), \\ h : (\Pi b : A)(\Pi s : C(a, i, b))M(b, r(b, s)) \\ \vdash q_2(a, i, r, h) : M(a, \text{tr}(a, i, r)) \end{array}}{\text{El}_{\triangleleft}(\text{tr}(a, i, r), q_1, q_2) = q_2(a, i, r, \lambda b. \lambda p. \text{El}_{\triangleleft}(r(b, p), q_1, q_2)) : M(a, \text{tr}(a, i, r))} \text{C}_{\text{tr}} - \triangleleft$$

Rules for well-founded predicates in mTT

$$\frac{\begin{array}{l} I \text{ set} \\ i : I \vdash N(i) \text{ set} \\ i : I, n : N(i) \vdash R(i, n) : I \rightarrow \text{prop}_s \end{array}}{i : I \vdash \text{WP}_R(i) \text{ prop}_s} \text{F-WP}$$

$$\begin{array}{c}
 \frac{i : I \quad n : N(i) \quad r : (\forall j : I)(R(i, n, j) \Rightarrow \text{WP}_R(j))}{\text{ind}(i, n, r) : \text{WP}_R(i)} \text{ I-WP} \\
 \\
 \frac{i : I \vdash P(i) \text{ prop} \quad i : I \quad p : \text{WP}_R(i) \quad i : I, n : N(i), s : (\forall j : I)(R(i, n, j) \Rightarrow P(j)) \vdash c(i, n, s) : P(i)}{\text{El}_{\text{WP}}(i, p, c) : P(i)} \text{ E-WP} \\
 \\
 \frac{i : I \vdash P(i) \text{ prop} \quad i : I \quad n : N(i) \quad r : (\forall j : I)(R(i, n, j) \Rightarrow \text{WP}_R(j)) \quad i : I, n : N(i), s : (\forall j : I)(R(i, n, j) \Rightarrow P(j)) \vdash c(i, n, s) : P(i)}{\text{El}_{\text{WP}}(i, \text{ind}(i, n, r), c) = c(i, n, \lambda_{\forall j}. \lambda_{\Rightarrow s}. \text{El}_{\text{WP}}(j, r(j, s), c)) : P(i)} \text{ C-WP}
 \end{array}$$

Rules for well-founded predicates in Martin-Löf's type theory

$$\begin{array}{c}
 I : \mathbf{U}_0 \\
 i : I \vdash N(i) : \mathbf{U}_0 \\
 \frac{i : I, n : N(i) \vdash R(i, n) : I \rightarrow \mathbf{U}_0}{i : I \vdash \text{WP}_R(i) : \mathbf{U}_0} \text{ F-WP} \\
 \\
 \frac{i : I \quad n : N(i) \quad f : (\Pi j : I)(R(i, n, j) \rightarrow \text{WP}_R(j))}{\text{ind}(i, n, f) : \text{WP}_R(i)} \text{ I-WP} \\
 \\
 \frac{i : I, w : \text{WP}_R(i) \vdash M(i, w) \text{ set} \quad i : I \quad w : \text{WP}_R(i) \quad i : I, \\
 n : N(i), \\
 f : (\Pi j : I)(R(i, n, j) \rightarrow \text{WP}_R(j)), \\
 h : (\Pi j : I)(\Pi r : R(i, n, j))M(j, f(j, r)) \\
 \vdash c(i, n, f, h) : M(i, \text{ind}(i, n, f))}{\text{El}_{\text{WP}}(i, w, c) : M(i, w)} \text{ E-WP} \\
 \\
 \frac{i : I, w : \text{WP}_R(i) \vdash M(i, w) \text{ set} \quad i : I \quad n : N(i) \quad f : (\Pi j : I)(R(i, n, j) \rightarrow \text{WP}_R(j)) \quad i : I, \\
 n : N(i), \\
 f : (\Pi j : I)(R(i, n, j) \rightarrow \text{WP}_R(j)), \\
 h : (\Pi j : I)(\Pi r : R(i, n, j))M(j, f(j, r)) \\
 \vdash c(i, n, f, h) : M(i, \text{ind}(i, n, f))}{\text{El}_{\text{WP}}(i, \text{ind}(i, n, f), c) = c(i, n, f, \lambda j. \lambda r. \text{El}_{\text{WP}}(c, j, f(j, r))) : M(i, \text{ind}(i, n, f))} \text{ C-WP}
 \end{array}$$