Combining Deduction Modulo and Logics of Fixed-Point Definitions

David Baelde
IT University of Copenhagen

Gopalan Nadathur
University of Minnesota

***Abstract*—Inductive and coinductive specifications are widely used in formalizing computational systems. Such specifications have a natural rendition in logics that support fixed-point definitions. Another useful formalization device is that of recursive specifications. These specifications are not directly complemented by fixed-point reasoning techniques and, correspondingly, do not have to satisfy strong monotonicity restrictions. We show how to incorporate a rewriting capability into logics of fixed-point definitions towards additionally supporting recursive specifications. In particular, we describe a natural deduction calculus that adds a form of "closed-world" equality—a key ingredient to supporting fixed-point definitions—to *deduction modulo*, a framework for extending a logic with a rewriting layer operating on formulas. We show that our calculus enjoys strong normalizability when the rewrite system satisfies general properties and we demonstrate its usefulness in specifying and reasoning about syntax-based descriptions. The integration of closed-world equality into deduction modulo leads us to reconfigure the elimination principle for this form of equality in a way that resolves long-standing issues concerning the stability of finite proofs under proof reduction.**

I. Introduction

Fixed-point definitions constitute a widely used specification device in computational settings. The process of reasoning about such definitions can be formalized within a logic by including a proof rule for introducing predicates from their definition, and a case analysis rule for eliminating such predicates in favor of the definitions through which they might have been derived. For example, given the following definition of natural numbers

$$nat\ 0 \stackrel{\triangle}{=} \top \qquad nat\ (s\ x) \stackrel{\triangle}{=} nat\ x$$

the introduction and elimination rules would respectively build in the capabilities of recognizing natural numbers and of reasoning by case analysis over them. When definitional clauses are positive, they are guaranteed to admit a fixed point and the logic can be proved to be consistent. Further, least (resp. greatest) fixed points can be characterized by adding an induction (resp. coinduction) rule to the logic. These kinds of treatments have been added to second-order logic [1], [2], type theory [3] and first-order logics [4]–[7].

The case analysis rule, which corresponds under the Curry-Howard isomorphism to pattern matching in computations, is complex in many formulations of the above ideas, and the (co)induction rules are even more so. By identifying and utilizing a suitable notion of equality, it is possible to give these rules a simple and elegant rendition. For example, the two clauses for *nat* can be transformed into the following form:

$$nat\ x \stackrel{\triangle}{=} x = 0 \lor \exists y.\ x = s\ y \land nat\ y$$

The case analysis rule can then be derived by unfolding a *nat* hypothesis into its single defining clause and using elimination rules for disjunction and equality. However, to obtain the expected behavior, equality elimination has to internalize aspects of term equality such as disjointness of constructors; e.g., the 0 branch should be closed immediately if the instantiation of x has the form $s\ n$. The introduction of this separate notion of equality, which we refer to as *closed-world equality*, has been central to the concise formulation of generic (co)induction rules [7]. Further, fixed-point combinators can be introduced to make the structure of (co)inductive predicates explicit rather than relying on a side table of definitions. Thus, the (inductive) definition of natural numbers may simply be rendered as $\mu\ (\lambda N \lambda x.\ x = 0 \lor \exists y.\ x = s\ y \land N\ y)$. Fixed point combinators simplify and generalize the theory, notably enabling mutual (co)induction schemes from the natural (co)induction rules [8], [9]. The logics resulting from this line of work, which we refer to as logics of fixed-point definitions from now on, have a simple structure that is well-adapted to automated and interactive proof-search [10], [11]. Moreover, they can be combined with features such as generic quantification that are useful in capturing binding structure to yield calculi that are well-suited to formalizing the meta-theory of computational and logical systems [12]–[14].

Logics featuring (co)inductive definitions can be made more powerful by adding another genre of definitions: recursive definitions based on inductive sets. A motivating context for such definitions is provided by the Tait-style strong normalizability argument [15], which figures often in the meta-theory of computational systems. For the simply typed λ-calculus, this argument relies on a *reducibility* relation specified by the following clauses:

$$\begin{aligned} red\ \iota\ e &\stackrel{\triangle}{=} sn\ e \\ red\ (t_1 \to t_2)\ e &\stackrel{\triangle}{=} \forall e'.\ red\ t_1\ e' \supset red\ t_2\ (e\ e') \end{aligned}$$

We assume ι to be the sole atomic type here and that *sn* is a predicate that recognizes strong normalizability. The specification of *red* looks deceptively like a fixed-point definition. However, treating it as such is problematic because the second clause in the definition does not satisfy the positivity condition. More importantly, the Tait-style argument does not involve reasoning on *red* like we reason on fixed-point definitions. Instead of performing case-analysis or induction on *red*, properties are proved about it using an (external) induction on types and the clauses for *red* mainly support an unfolding of the definition once the structure of a type is known [16]. Generally, recursive definitions are distinguished by the fact that they embody computations or rewriting within proofs rather than the case analysis and speculative rewriting that is characteristic of fixed-point based reasoning.

In this paper, we show how to incorporate the capability of recursive definitions into logics of fixed-point definitions. At a technical level, we do this by introducing least and greatest fixed points and the idea of closed-world equality into *deduction modulo* [17], a framework for extending a logic with a rewriting layer that operates on formulas and terms. This rewriting layer allows for a transparent treatment of recursive definitions, but a satisfactory encoding of closed-world equality (and thus fixed-point definitions) seems outside its reach. This dichotomy actually highlights the different strengths of logics of fixed-point definitions and deduction modulo: while the former constitute excellent vehicles for dealing with (co)inductive definitions, the rewriting capability of the latter is ideally suited for supporting recursive definitions. By extending deduction modulo with closed-world equality and fixed points, we achieve a combination of these strengths. This combination also clarifies the status of our equality: we show that it is compatible with a theory on terms and is thus richer than a simple "syntactic" form of equality.

The main technical result of this paper is a strong normalizability property for our enriched version of deduction modulo. The seminal work in this context is that of Dowek and Werner [18], who provide a proof of strong normalizability for deduction modulo that is modular with respect to the rewriting system being used. In the course of adapting this proof to our setting, we rework previous logical treatments of closed-world equality in a way that, for the first time, lets us require that proofs be finite without sacrificing their stability under reduction. For the resulting system, we are able to construct a proof of strong normalizability which follows very naturally the intended semantics of fixed-point and recursive definitions: the former are interpreted as a whole using a semantic fixed-point, while the latter are interpreted instance by instance. Regarding the normalization of least and greatest fixed-point constructs, our work adapts that of Baelde [9] from linear to intuitionistic logic. We use a natural deduction style in presenting our logic that has the virtue of facilitating future investigations of connections with functional programming.

The rest of the paper is structured as follows. In Section II, we motivate and present our logical system. Section III describes reductions on proofs. Section IV provides a proof of strong normalizability that is modular in the rewrite rules being considered. We use this result to facilitate recursive definitions in Section V and we illustrate their use in formalizing the meta-theory of programming languages. Section VI discusses related and future work.

II. Deduction Modulo with Fixed-Points and Equality

We present our extension to deduction modulo in the form of a typing calculus for appropriately structured proof terms. This gives us a convenient tool for defining proof reductions and proving strong normalizability in later sections.

A. Formalizing closed-world equality

We first provide an intuition into our formalization of the desired form of equality. The rule for introducing an equality is the expected one: two terms are equal if they are congruent modulo the operative rewriting relation. Denoting the congruence by $\equiv$, this rule can simply be

$$\frac{}{\Gamma \vdash t = t'}\ t \equiv t'$$

The novelty is in the elimination rule that must encapsulate the closed-world interpretation. This can be captured in the form of a case analysis over all unifiers of the eliminated equality; the unifiers that are relevant to consider here would instantiate variables of universal strength, called eigenvariables, in the terms. One formulation of this idea that has been commonly used in the literature is the following:

$$\frac{\Gamma \vdash t = t' \quad \{\ \Gamma\theta_i \vdash P\theta_i \mid \theta_i \in csu(t,t')\ \}}{\Gamma \vdash P}$$

The notation $csu(t,t')$ is used here to denote a *complete set of unifiers* for t and t' modulo $\equiv$, i.e., a set of unifiers such that every unifier for the two terms is subsumed by a member of the set. The closed world assumption is expressed in the fact that $\Gamma \vdash P$ needs to be proved under only these substitutions. Note in particular that the set of right premises here is empty when t and t' are not unifiable, i.e., have no common instances.

The equality left rule could have simply used the set of *all* unifiers for t and t'. Basing it on *csu*s instead allows the cardinality of the premise set to be controlled, typically permitting it to be reduced to a finite collection from an infinite one. However, a problem with the way this rule is formulated is that this property is not stable under substitution. For example, consider the following derivation in which x and y are variables:

$$\frac{p\ x, x = y \vdash x = y \quad (p\ x, x = y)[y/x] \vdash (p\ y)[y/x]}{p\ x, x = y \vdash p\ y}$$

If we were to apply the substitution $[t_1/x, t_2/y]$ to it, the branching structure of the derivation would have to be changed to reflect the nature of a *csu* for t_1 and t_2; this could well be an infinite set. A related problem manifests itself when we need to substitute a proof π for an assumption into the derivation. If we were to work the proof substitution eagerly through each of the premises in the equality elimination rule, it would be necessary to modify the structure of π to accord with the term substitution that indexes each of the premise derivations. In the context of deduction modulo, the instantiation in π can create new opportunities for rewriting formulas. Since the choice of the "right" premise cannot be determined upfront, the eager propagation of proof substitutions into equality eliminations can lead to a form of speculative rewriting which, as we shall see, is problematic when recursive definitions are included.

We avoid these problems by formulating equality elimination in a way that allows for the *suspension* of term and proof substitutions. Specifically, this rule is

$$\frac{\Gamma' \vdash t\theta = t'\theta \quad \Gamma' \vdash \Gamma\theta \quad \{\, \Gamma\theta_i \vdash P\theta_i \mid \theta_i \in csu(t, t') \,\}}{\Gamma' \vdash P\theta}$$

Here, $\Gamma' \vdash \Gamma\theta$ means that there is a derivation of $\Gamma' \vdash Q$ for any $Q \in \Gamma\theta$. This premise, that introduces a form of *cut*, allows to delay the propagation of proof substitutions over the premises that represent the case analysis part of the rule. Notice also that we consider *csu*s for t and t' and not $t\theta$ and $t'\theta$ over these premises, i.e., the application of the substitution θ is also suspended. Of course, these substitutions must eventually be applied. Forcing the application becomes the task of the reduction rule for equality that also simultaneously selects the right branch in the case analysis.

Our equality elimination rule also has the pleasing property of allowing the structure of proofs to be preserved under substitutions. For example, the proof

$$\frac{\overline{p\ x, x = y \vdash x = y} \quad \overline{p\ x, x = y \vdash p\ x} \quad \overline{(p\ x)[y/x] \vdash (p\ y)[y/x]}}{p\ x, x = y \vdash p\ y}$$

under the substitution $\theta := [t_1/x, t_2/y]$ becomes

$$\frac{\overline{\Gamma \vdash (x = y)\theta} \quad \overline{\Gamma \vdash (p\ x)\theta} \quad \overline{(p\ x)[y/x] \vdash (p\ y)[y/x]}}{\Gamma \vdash p\ t_2}$$

where $\Gamma = (\ p\ t_1,\ t_1 = t_2\ )$.

B. The logic μNJ modulo

The syntax of our formulas is based on a language of typed λ-terms. We do not describe these in detail and assume only that it is equipped with standard notions of variables and substitutions. We distinguish o as the type of propositions. Term types, denoted by γ, are ones that do not contain o. Predicates are expressions of type $\gamma_1 \to \ldots \to \gamma_n \to o$. Both formulas and predicates are denoted by P or Q. We use p or q for predicate variables and a for predicate constants. Terms are expressions of term types, and shall be denoted by t, u or v. We use x, y or z for term variables. All expressions are considered up-to β- and η-conversion. In addition to that basic syntactic equality, we assume a congruence relation $\equiv$. In Section V, we will describe sufficient conditions on that congruence for ensuring consistency of the logic.

Definition 1. *A unifier of u and v is a substitution θ such that $u\theta \equiv v\theta$. A* complete set of unifiers *for u and v, written* $\mathrm{csu}(u, v)$ *is a set $\{\ \theta_i\ \}_i$ of unifiers of u and v, such that any other unifier of u and v is of the form $\theta_i\theta'$ for some i and θ'. Note that complete sets of unifiers are not unique. However, this ambiguity will be harmless in our setting.*

Definition 2. *Formulas are built as follows:*

$$\begin{array}{rcl} P & ::= & \top \mid \bot \mid P \supset Q \mid P \wedge Q \mid P \vee Q \mid \forall x.P \mid \exists x.P \mid \\ & & t = t' \mid (\mu\ B\ \vec{t}) \mid (\nu\ B\ \vec{t}) \mid (p\ \vec{t}) \mid (a\ \vec{t}) \end{array}$$

Here, $\wedge$, $\vee$ and $\supset$ are connectives of type $o \to o \to o$, equality has type $\gamma \to \gamma \to o$ and quantifiers have type $(\gamma \to o) \to o$ for any γ. Expressions of the form $a\,\vec{t}$ are called atomic formulas. *The least and greatest fixed point combinators μ and ν have the type $(\tau \to \tau) \to \tau$ for any τ of the form $\vec{\gamma} \to o$. The first argument for these combinators, denoted by B, must have the form $\lambda p \lambda \vec{x}.P$ called a predicate operator. Every predicate variable occurrence must be within such an operator, bound by the first abstraction in it. An occurrence of p in a formula is* positive *if it is on the left of an even number of implications, and it is* negative *otherwise and $\lambda p \lambda \vec{x}.P$ is said to be monotonic (resp. antimonotonic) if p occurs only positively (resp. negatively) in P. We restrict the first argument of fixed-point combinators to be monotonic operators.*

We now introduce a language of proof terms, and define type assignment. The constructs and typing rules are standard (e.g., see [18]), with the exception of those for equality and fixed points. Following the Curry-Howard correspondence, (proof-level) types correspond to formulas, typing derivations correspond to proofs, and the reduction of proof terms corresponds to proof normalization. The guidelines determining the form of the new proof terms are that all information needed for reduction should be included in them and that type checking should be easily decidable. The details of our choices should become clear when we present the typing rules.

Definition 3. *Proof terms, denoted by π and ρ, are given by the following syntax rules:*

$$\begin{array}{rcl} \pi & ::= & \alpha \mid \langle\rangle \mid \delta_\bot(\pi) \\ & \mid & \lambda\alpha.\pi \mid (\pi\ \pi') \\ & \mid & \langle \pi, \pi' \rangle \mid \mathrm{proj}_1(\pi) \mid \mathrm{proj}_2(\pi) \\ & \mid & \mathrm{in}_1(\pi) \mid \mathrm{in}_2(\pi) \mid \delta_\vee(\pi_1, \alpha.\pi_2, \beta.\pi_3) \\ & \mid & \lambda x.\pi \mid (\pi\ t) \\ & \mid & \langle t, \pi \rangle \mid \delta_\exists(\pi, x.\alpha.\pi') \\ & \mid & \mathrm{refl} \mid \delta_=(\Gamma, \theta, \sigma, u, v, P, \pi, (\theta'_i.\pi_i)_i) \\ & \mid & \mu(B, \vec{t}, \pi) \mid \delta_\mu(\pi, \vec{x}.\alpha.\pi') \\ & \mid & \nu(\pi, \alpha.\pi') \mid \delta_\nu(B, \vec{t}, \pi) \end{array}$$

Here and later, we use α, β, γ to denote proof variables, and σ to denote substitutions for proof variables. The notation $(\theta'_i.\pi_i)_i$ in the equality elimination construct stands for an arbitrary number of subterms; it may be empty, but it must always be finite. In the expression $\theta.\pi$, all free variables of π must be in the range of the substitution θ — this is not restrictive since θ may be extended at will with identity substitutions. Finally, the notation $x.\pi$ or $\alpha.\pi$ denotes a binding construct, i.e., x (resp. α) is bound in π. As usual, terms are identified up to renaming of bound variables, and renaming is used to avoid capture when propagating a substitution under a binder.

Definition 4. *A proof term π has type P under the context Γ if $\Gamma \vdash \pi : P$ is derivable using the rules in Figure 1. We also say that $\Gamma' \vdash \sigma : \Gamma$ holds if Γ and σ have the same domain and $\Gamma' \vdash \sigma(\alpha) : \Gamma(\alpha)$ holds for each α in that domain.*

$$\frac{}{\Gamma \vdash \alpha : P}\ P \equiv Q,\ (\alpha : Q) \in \Gamma \qquad \frac{}{\Gamma \vdash \langle\rangle : P}\ P \equiv \top \qquad \frac{\Gamma \vdash \pi : \bot}{\Gamma \vdash \delta_{\bot}(\pi) : P}$$

$$\frac{\Gamma, \alpha : P_1 \vdash \pi : P_2}{\Gamma \vdash \lambda\alpha.\pi : P}\ P \equiv P_1 \supset P_2 \qquad \frac{\Gamma \vdash \pi : Q \supset P \quad \Gamma \vdash \pi' : Q}{\Gamma \vdash \pi\,\pi' : P}$$

$$\frac{\Gamma \vdash \pi_1 : P_1 \quad \Gamma \vdash \pi_2 : P_2}{\Gamma \vdash \langle \pi_1, \pi_2 \rangle : P}\ P \equiv P_1 \wedge P_2 \qquad \frac{\Gamma \vdash \pi : P_1 \wedge P_2}{\Gamma \vdash proj_i(\pi) : P'_i}\ P'_i \equiv P_i,\ i \in \{1, 2\}$$

$$\frac{\Gamma \vdash \pi : P_i}{\Gamma \vdash in_i(\pi) : P}\ P \equiv P_1 \vee P_2 \qquad \frac{\Gamma \vdash \pi : P_1 \vee P_2 \quad \Gamma, \alpha : P_1 \vdash \pi_1 : P \quad \Gamma, \beta : P_2 \vdash \pi_2 : P}{\Gamma \vdash \delta_{\vee}(\pi, \alpha.\pi_1, \beta.\pi_2) : P}$$

$$\frac{\Gamma \vdash \pi : Q}{\Gamma \vdash \lambda x.\pi : P}\ P \equiv \forall x.Q \qquad \frac{\Gamma \vdash \pi : \forall x.Q}{\Gamma \vdash \pi\, t : P}\ P \equiv Q[t/x]$$

$$\frac{\Gamma \vdash \pi : Q[t/x]}{\Gamma \vdash \langle t, \pi \rangle : P}\ P \equiv \exists x.Q \qquad \frac{\Gamma \vdash \pi : \exists x.Q \quad \Gamma, \alpha : Q \vdash \pi' : P}{\Gamma \vdash \delta_{\exists}(\pi, x.\alpha.\pi') : P}$$

$$\frac{}{\Gamma \vdash refl : P}\ P \equiv (t = t) \qquad \frac{\Gamma' \vdash \pi : u\theta = v\theta \quad \Gamma' \vdash \sigma : \Gamma\theta \quad (\Gamma\theta'_i \vdash \pi_i : Q\theta'_i)_i}{\Gamma' \vdash \delta_{=}(\Gamma, \theta, \sigma, u, v, Q, \pi, (\theta'_i.\pi_i)_i) : P}\ (\theta'_i)_i \in csu(u, v), P \equiv Q\theta$$

$$\frac{\Gamma \vdash \pi : B\,(\mu B)\,\vec{t}}{\Gamma \vdash \mu(B, \vec{t}, \pi) : P}\ P \equiv \mu\, B\, \vec{t} \qquad \frac{\Gamma \vdash \pi : \mu\, B\, \vec{t} \quad \Gamma, \alpha : B\, S\, \vec{x} \vdash \pi' : S\, \vec{x}}{\Gamma \vdash \delta_{\mu}(\pi, \vec{x}.\alpha.\pi') : P}\ P \equiv S\, \vec{t}$$

$$\frac{\Gamma \vdash \pi : S\, \vec{t} \quad \Gamma, \alpha : S\, \vec{x} \vdash \pi' : B\, S\, \vec{x}}{\Gamma \vdash \nu(\pi, \vec{x}.\alpha.\pi') : P}\ P \equiv \nu\, B\, \vec{t} \qquad \frac{\Gamma \vdash \pi : \nu\, B\, \vec{t}}{\Gamma \vdash \delta_{\nu}(B, \vec{t}, \pi) : P}\ P \equiv B\,(\nu B)\,\vec{t}$$

Variables bound in proof terms are assumed to be new in instances of typing rules, i.e., they should not occur free in the base sequent. Specifically, α, β, x are assumed to be new in the introduction rules for implication, universal quantification and greatest fixed-point, as well as elimination rules for disjunction, existential quantification, equality and least fixed-point.

Fig. 1: μNJ: Natural deduction modulo with equality and least and greatest fixed points

C. Expressiveness of the logic

The logic μNJ modulo inherits from logics of fixed-point definitions a simplicity in the treatment of (co)inductive sets and relations and from deduction modulo the ability to blend computation and deduction in the course of reasoning. We illustrate this aspect through a few simple examples here.

Natural numbers may be specified through the following least fixed point predicate:

$$nat \stackrel{def}{=} \mu(\lambda N \lambda x.\ x = 0 \vee \exists y.\ x = s\ y \wedge N\ y)$$

Specialized for this predicate, the least fixed point rules immediately give rise to the following standard derived rules:

$$\frac{}{\Gamma \vdash nat\ 0} \qquad \frac{\Gamma \vdash nat\ x}{\Gamma \vdash nat\ (s\ x)}$$

$$\frac{\Gamma \vdash nat\ x \quad \Gamma \vdash P\ 0 \quad \Gamma, P\ y \vdash P\ (s\ y)}{\Gamma \vdash P\ x}\ y \text{ new}$$

Having natural numbers, we can easily obtain the rest of Heyting arithmetic. Addition may be defined as an inductive relation, but the congruence also allows it to be defined more naturally as a term-level function, equipped with the rewrite rules $0 + y \rightsquigarrow y$ and $(s\ x) + y \rightsquigarrow s\ (x + y)$. Treating it in the latter way allows us to exploit the standard dichotomy between deduction and computation in deduction modulo to shorten proofs [19]. For example, $(s\ 0) + (s\ 0) = s\ (s\ 0)$ can be proved in one step by using the fact that the two terms in the equation are congruent to each other. More general properties about addition defined in this way must be conditioned by assumptions about the structure of the terms. For instance, commutativity of addition should be stated as follows:

$$\forall x \forall y.\ nat\ x \supset nat\ y \supset x + y = y + x$$

This theorem is proved by induction on the *nat* hypotheses, with the computation of addition being performed implicitly in the congruence when the structure of the first summand becomes known. Note that we do not have to know how to compute *csu*s modulo arithmetic to build that derivation: all that is needed is the substitutivity principle $\forall x \forall y.\ x = y \supset P\ x \supset P\ y$ which only involves shallow unification.

III. Reductions on Proof Terms

As usual, we consider reducing proof terms in which an elimination rule for a logical symbol immediately follows an introduction rule for the same symbol. Substitutions for both term-level and proof-level variables play an important role in describing such reductions. They are defined as usual, extended as shown on Figure 2 for equality and for the least and greatest fixed-point constructs. Note that substitutions are suspended over the parts representing case analysis in the equality elimination rule as discussed earlier. The next two lemmas show that this treatment of substitution is coherent.

Lemma 1. *Term-level substitution preserves type assignment:* $\Gamma \vdash \pi : P$ *implies* $\Gamma\theta \vdash \pi\theta : P\theta$.

$$(\delta_=(\Gamma,\theta',\sigma,u,v,P,\pi,(\theta''_i.\pi_i)_i))\theta \stackrel{def}{=} \delta_=(\Gamma,\theta'\theta,\sigma\theta,u,v,P,\pi\theta,(\theta''_i.\pi_i)_i)$$
$$(\delta_=(\Gamma,\theta',\sigma',u,v,P,\pi,(\theta''_i.\pi_i)_i))\sigma \stackrel{def}{=} \delta_=(\Gamma,\theta',\sigma'\sigma,u,v,P,\pi\sigma,(\theta''_i.\pi_i)_i)$$

$$(\mu(B,\vec{t},\pi))\theta \stackrel{def}{=} \mu(B\theta,\vec{t}\theta,\pi\theta) \qquad (\delta_\mu(\pi,\vec{x}.\alpha.\pi'))\theta \stackrel{def}{=} \delta_\mu(\pi\theta,\vec{x}.\alpha.\pi'\theta)$$
$$(\mu(B,\vec{t},\pi))\sigma \stackrel{def}{=} \mu(B,\vec{t},\pi\sigma) \qquad (\delta_\mu(\pi,\vec{x}.\alpha.\pi'))\sigma \stackrel{def}{=} \delta_\mu(\pi\sigma,\vec{x}.\alpha.\pi'\sigma)$$

$$(\nu(\pi,\vec{x}.\alpha.\pi'))\theta \stackrel{def}{=} \nu(\pi\theta,\vec{x}.\alpha.\pi'\theta) \qquad (\delta_\nu(B,\vec{t},\pi))\theta \stackrel{def}{=} \delta_\nu(B\theta,\vec{t}\theta,\pi\theta)$$
$$(\nu(\pi,\vec{x}.\alpha.\pi'))\sigma \stackrel{def}{=} \nu(\pi\sigma,\vec{x}.\alpha.\pi'\sigma) \qquad (\delta_\nu(B,\vec{t},\pi))\sigma \stackrel{def}{=} \delta_\nu(B,\vec{t},\pi\sigma)$$

Fig. 2: Term and proof-level substitutions into equality, least and greatest fixed-point proof terms

Proof: This is easily checked by induction on the typing derivation. An interesting case is that of equality elimination. Consider the following derivation:

$$\frac{\Gamma' \vdash \pi : u\theta' = v\theta' \quad \Gamma' \vdash \sigma : \Gamma\theta' \quad (\Gamma\theta''_i \vdash \pi_i : P'\theta''_i)_i}{\Gamma' \vdash \delta_=(\Gamma,\theta',\sigma,u,v,P',\pi,(\theta''_i.\pi_i)_i) : P}\ P \equiv P'\theta'$$

By the induction hypothesis, $\Gamma'\theta \vdash \pi\theta : u\theta'\theta = v\theta'\theta$ and $\Gamma'\theta \vdash \sigma\theta : \Gamma\theta'\theta$ have derivations. From these we build the derivation

$$\frac{\Gamma'\theta \vdash \pi\theta : u\theta'\theta = v\theta'\theta \quad \Gamma'\theta \vdash \sigma\theta : \Gamma\theta'\theta \quad (\Gamma\theta''_i \vdash \pi_i : P'\theta''_i)_i}{\Gamma'\theta \vdash \delta_=(\Gamma,\theta'\theta,\sigma\theta,u,v,P',\pi\theta,(\theta''_i.\pi_i)_i) : P\theta}$$

■

Lemma 2. *If* $\Gamma \vdash \pi : P$ *and* $\Gamma' \vdash \sigma : \Gamma$ *then* $\Gamma' \vdash \pi\sigma : P$.

Proof: This is shown also by induction on the typing derivation. An interesting case, again, is that of equality elimination. Consider the following derivation:

$$\frac{\Gamma \vdash \pi : u\theta = v\theta \quad \Gamma \vdash \sigma' : \Gamma''\theta \quad (\Gamma''\theta'_i \vdash \pi_i : P'\theta'_i)_i}{\Gamma \vdash \delta_=(\Gamma'',\theta,\sigma',u,v,P',\pi,(\theta'_i.\pi_i)_i) : P}\ P \equiv P'\theta$$

By the induction hypothesis, $\Gamma' \vdash \pi\sigma : u\theta = v\theta$ and $\Gamma' \vdash \sigma'\sigma : \Gamma''\theta$ have derivations. From this we build the derivation

$$\frac{\Gamma' \vdash \pi\sigma : u\theta = v\theta \quad \Gamma' \vdash \sigma'\sigma : \Gamma''\theta \quad (\Gamma''\theta'_i \vdash \pi_i : P'\theta'_i)_i}{\Gamma' \vdash \delta_=(\Gamma'',\theta,\sigma'\sigma,u,v,P',\pi\sigma,(\theta'_i.\pi_i)_i) : P}$$

■

The most interesting reduction rules are those for the least and greatest fixed-point operators. In the former case, the rule must apply to a proof of the form

$$\frac{\dfrac{\Gamma \vdash \pi : B\ (\mu\ B)\ \vec{t}}{\Gamma \vdash \mu(B,\vec{t},\pi) : \mu\ B\ \vec{t}} \quad \Gamma,\alpha : B\ S\ \vec{x} \vdash \pi' : S\ \vec{x}}{\Gamma \vdash \delta_\mu(\mu(B,\vec{t},\pi),\vec{x}.\alpha.\pi') : S\ \vec{t}}$$

This redex can be eliminated by generating a proof of $\Gamma \vdash S\ \vec{t}$ directly from the derivation of $\Gamma \vdash \pi : B\ (\mu\ B)\ \vec{t}$: doing this effectively means that we move the redex (cut) deeper into the iteration that introduces the least fixed point. To realize this transformation, we proceed as follows:

- Using the derivation π', we can get a proof of $S\ \vec{t}$ from $B\ S\ \vec{t}$. Thus, the task reduces to generating a proof of $B\ S\ \vec{t}$ from $B\ (\mu\ B)\ \vec{t}$.
- Using again π', we get a derivation for $\Gamma, \beta : \mu\ B\ \vec{x} \vdash \delta_\mu(\beta,\vec{x}.\alpha.\pi') : S\ \vec{x}$. If we can show how to "lift" this derivation over the operator $\lambda p.(B\ p\ \vec{t})$, we obtain the needed derivation of $B\ S\ \vec{t}$ from $\pi : B\ (\mu\ B)\ \vec{t}$.

For the latter step, we use the notion of *functoriality* [2]. For any monotonic operator B, we define the functor F_B for which the following typing rule is admissible:

$$\frac{\Gamma,\alpha : P\vec{x} \vdash \pi : P'\vec{x}}{\Gamma \vdash F_B(\vec{x}.\alpha.\pi) : (B\ P) \supset (B\ P')}$$

Definition 5 (Functoriality, $F_B(\pi)$)**.** *Let B be an operator of type* $(\vec{\gamma} \to o) \to o$*, and* π *be a proof such that* $\alpha : P\ \vec{x} \vdash \pi : P'\ \vec{x}$*. We define* $F^+_B(\vec{x}.\alpha.\pi)$ *of type* $B\ P \supset B\ P'$ *for a monotonic B and* $F^-_B(\vec{x}.\alpha.\pi)$ *of type* $B\ P' \supset B\ P$ *for an antimonotonic B by induction on the maximum depth of an occurrence of p in B p through the rules in Figure 3. In these rules,* $*$ *denotes any polarity (+ or −) and* $-*$ *denotes the complementary one. We write* $F^+_B(\vec{x}.\alpha.\pi)$ *more simply as* $F_B(\vec{x}.\alpha.\pi)$*.*

Checking the admissibility of the typing rule pertaining to F_B is mostly routine. We illustrate how this is to be done by considering the least fixed point case in Figure 4; the greatest fixed point case is shown in Figure 7 in the appendix.

The full collection of reduction rules is presented in Figure 5. Note that the reduction rule for equality is not deterministic as stated: determinism can be forced if needed by suitable assumptions on *csu*s or by forcing a particular choice of θ'_i and θ'' in case of multiple possibilities.

Theorem 1 (Subject reduction)**.** *If* $\Gamma \vdash \pi : P$ *and* $\pi \to \pi'$ *then* $\Gamma \vdash \pi' : P$.

Proof: This follows from the above substitution lemmas. For example, consider the equality case. If $u\theta \equiv v\theta$ then $\delta_=(\Gamma',\theta,\sigma,u,v,P,\mathit{refl},(\theta'_i.\pi'_i)_i) \to \pi'_i\theta''\sigma$ where $\theta = \theta'_i\theta''$. We have a derivation of $\Gamma'\theta'_i \vdash \pi'_i : P\theta'_i$. Hence, by applying θ'' and using Lemma 1, $\Gamma'\theta \vdash \pi'_i\theta'' : P\theta$ must have a derivation. Finally, since $\Gamma \vdash \sigma : \Gamma'\theta$ has a derivation, by Lemma 2 there must be one for $\Gamma \vdash \pi'_i\theta''\sigma : P\theta$. ■

Proposition 1. *For any proof terms* π*,* π' *and* ρ *and any term t,* $\pi \to \pi'$ *implies* $\pi[\rho/\alpha] \to \pi'[\rho/\alpha]$ *and* $\pi \to \pi'$ *implies* $\pi[t/x] \to \pi'[t/x]$.

Proof: Both statements are easily checked. ■

A proof is said to be normal if there are no redexes in the proof term that represents it. One of the uses of this notion,

$$
\begin{array}{rcl}
F^{+}_{\lambda p.p\vec{t}}(\vec{x}.\alpha.\pi) = \lambda\alpha.\pi[\vec{t}/\vec{x}] & & F^{*}_{\lambda p.Q}(\vec{x}.\alpha.\pi) = \lambda\beta.\beta \text{ if } p \text{ does not occur in } Q \\
F^{*}_{\lambda p.(B_1\ p)\wedge(B_2\ p)}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\langle F^{*}_{B_1}(\vec{x}.\alpha.\pi)\ (proj_1(\beta)), F^{*}_{B_2}(\vec{x}.\alpha.\pi)\ (proj_2(\beta))\rangle \\
F^{*}_{\lambda p.(B_1\ p)\vee(B_2\ p)}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\delta_{\vee}(\beta, \gamma.in_1(F^{*}_{B_1}(\vec{x}.\alpha.\pi)\ \gamma), \gamma.in_2(F^{*}_{B_2}(\vec{x}.\alpha.\pi)\ \gamma)) \\
F^{*}_{\lambda p.(B_1\ p)\supset(B_2\ p)}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\lambda\gamma.F^{*}_{B_2}(\vec{x}.\alpha.\pi)\ (\beta\ (F^{-*}_{B_1}(\vec{x}.\alpha.\pi)\ \gamma)) \\
F^{*}_{\lambda p.\forall x.(B\,p\,x)}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\lambda x.F^{*}_{\lambda p.B\,p\,x}(\vec{x}.\alpha.\pi)\ (\beta\ x) \\
F^{*}_{\lambda p.\exists x.(B\,p\,x)}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\delta_{\exists}(\beta, x.\gamma.\langle x, F^{*}_{\lambda p.B\,p\,x}(\vec{x}.\alpha.\pi)\ \gamma\rangle) \\
F^{*}_{\lambda p.\mu\,(B\ p)\,\vec{t}}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\delta_{\mu}(\beta, \vec{x}.\gamma.\mu(B\ P', \vec{x}, F^{*}_{\lambda p.B\ p\ (\mu\,(B\ P'))\,\vec{x}}(\vec{x}.\alpha.\pi)\ \gamma)) \\
F^{*}_{\lambda p.\nu\,(B\ p)\,\vec{t}}(\vec{x}.\alpha.\pi) & = & \lambda\beta.\nu(\beta, \vec{x}.\gamma.F^{*}_{(\lambda p.B\ p\ (\nu\,(B\ P))\,\vec{x})}(\vec{x}.\alpha.\pi)\ \delta_{\nu}(B\ P, \vec{x}, \gamma))
\end{array}
$$

Fig. 3: Definition of functoriality

$$
\dfrac{\dfrac{\dfrac{}{\Gamma,\beta:\mu\,(B\ P)\,\vec{t}\vdash\beta:\mu\,(B\ P)\,\vec{t}} \qquad \dfrac{\Gamma,\beta:\mu\,(B\ P)\,\vec{t},\gamma:B\ P\ (\mu\,(B\ P'))\ \vec{x}\vdash F_{\lambda p.B\ p\ (\mu\,(B\ P'))\ \vec{x}}(\vec{x}.\alpha.\pi)\ \gamma : B\ P'\ (\mu\,(B\ P'))\ \vec{x}}{\Gamma,\beta:\mu\,(B\ P)\,\vec{t},\gamma:B\ P\ (\mu\,(B\ P'))\ \vec{x}\vdash\mu(B\ P',\vec{x},\ldots):\mu\,(B\ P')\,\vec{x}}}{\Gamma,\beta:\mu\,(B\ P)\,\vec{t}\vdash\delta_{\mu}(\beta,\ldots):\mu\,(B\ P')\,\vec{t}}}{\Gamma\vdash F^{+}_{\lambda p.\mu\,(B\ p)\,\vec{t}}(\vec{x}.\alpha.\pi):\mu\,(B\ P)\,\vec{t}\supset\mu\,(B\ P')\,\vec{t}}
$$

Fig. 4: Typing functoriality for least fixed-points

$$
\begin{array}{c}
(\lambda\alpha.\pi)\ \pi' \ \rightarrow\ \pi[\pi'/\alpha] \qquad proj_i(\langle\pi_1,\pi_2\rangle)\ \rightarrow\ \pi_i \qquad \delta_{\vee}(in_i(\pi), \alpha.\pi_1, \alpha.\pi_2)\ \rightarrow\ \pi_i[\pi/\alpha] \\
(\lambda x.\pi)\ t\ \rightarrow\ \pi[t/x] \qquad \delta_{\exists}(\langle t,\pi\rangle, x.\alpha.\pi')\ \rightarrow\ \pi'[t/x][\pi/\alpha] \\
\delta_{\mu}(\mu(B,\vec{t},\pi), \vec{x}.\alpha.\pi')\ \rightarrow\ \pi'[\vec{t}/\vec{x}][(F_{\lambda p.B\,p\,\vec{t}}(\vec{x}.\beta.\delta_{\mu}(\beta, \vec{x}.\alpha.\pi'))\ \pi)/\alpha] \\
\delta_{\nu}(B,\vec{t},\nu(\pi, \vec{x}.\alpha.\pi'))\ \rightarrow\ F_{\lambda p.B\,p\,\vec{t}}(\vec{x}.\beta.\nu(\beta, \vec{x}.\alpha.\pi'))\ (\pi'[\vec{t}/\vec{x}][\pi/\alpha]) \\
\delta_{=}(\Gamma,\theta,\sigma,u,v,P,\mathit{refl},(\theta'_i.\pi_i)_i)\ \rightarrow\ \pi_i\theta''\sigma \text{ where } \theta = \theta'_i\theta''
\end{array}
$$

Fig. 5: Reduction rules for μNJ proof terms

and of the normalizability of proof terms, is in showing the consistency of a logic.

Lemma 3. *Provided that $\equiv$ is defined by a confluent rewrite system, rewriting terms to terms and atomic propositions to propositions, there is no normal proof of $\vdash \bot$.*

Proof: We first observe that typed normal forms are characterized as usual: no introduction term is ever found as the main parameter of an elimination. This standard property is not affected by our new constructs. For example, consider the case of equality: $\delta_{=}(\ldots, \mathit{refl}, (\theta_i.\pi_i)_i)$ can always be reduced by definition of complete sets of unifiers. The rest of the proof follows the usual lines: the proof cannot end with an elimination, otherwise it would have to be a chain of eliminations terminated with a proof variable, and there is no variable in the environment; it also cannot end with an introduction since there is no introduction for $\bot$ and the congruence cannot equate it with another connective. ∎

IV. Strong Normalizability

In a fashion similar to [18], we now establish strong normalizability for proof reductions when the congruence relation satisfies certain general conditions. The proof is based on the framework of reducibility candidates, and borrows elements from earlier work in linear logic [9] regarding fixed-points.

Definition 6. *A proof term is* neutral *iff it is not an introduction, i.e., it is a variable or an elimination construct.*

Definition 7. *A set R of proof terms is a* reducibility candidate *if (1) $R \subseteq \mathcal{SN}$; (2) $\pi \in R$ and $\pi \rightarrow \pi'$ implies $\pi' \in R$; and (3) if π is neutral and all of its one-step reducts are in R, then $\pi \in R$. We denote by C the set of all reducibility candidates.*

Note that conditions (2,3) are positive and compatible with (1) so that for any set of $\mathcal{SN}$ proofs there is a least candidate containing that set; let us call this operation *saturation*. Reducibility candidates, equipped with inclusion, form a complete lattice: the intersection of a family of candidates gives their infimum and the saturated union gives their supremum. Having a complete lattice, we can define least and greatest fixed points of monotonic operators. Finally, all this is lifted pointwise for functions from terms to candidates, which we call *predicate candidates*. We use $\mathcal{X}$ or $\mathcal{Y}$ to denote candidates and predicate candidates.

Definition 8. *A* pre-model *$\mathcal{M}$ consists in the interpretation of atomic formulas by reducibility candidates: for any predicate constant a of type $\gamma_1 \rightarrow \ldots \gamma_n \rightarrow o$, its interpretation $\hat{a}$ is a function from $|\gamma_1| \times \ldots \times |\gamma_n|$ to C, where $|\gamma|$ denotes the set of (potentially open) terms of type γ.*

Definition 9. *Let $\mathcal{M}$ be a pre-model. Let P be a formula and $\mathcal{E}$*

a context of candidates that covers all free predicate variables of P. We define the candidate $|P|^{\mathcal{E}}$*, called* interpretation of P, *by induction on the structure of P as shown in Figure 6.*

We now justify this definition, i.e., we show that $|P|^{\mathcal{E}}$ is always a candidate and that the fixed points formed in the interpretation actually exist. This is done by induction on P, simultaneously establishing that $|P|^{\mathcal{E}}$ is a candidate and that $|P|^{\mathcal{E}}$ is monotonic (resp. anti-monotonic) in $\mathcal{E}(p)$ for any variable p that only occurs positively (resp. negatively) in P. Except in the fixed point cases it is easy to check that (anti)monotonicity is preserved by our constructions (in the implication case, each statement follows from the other) and that the three conditions for being a reducibility candidate are obviously satisfied. The treatment of the two fixed point combinators is similar. We only detail the least fixed point case, i.e., $|\mu B\vec{t}|^{\mathcal{E}} = \mathit{lfp}(\phi)$. First, this is well-defined: by induction hypothesis on $Bp\vec{t}$, ϕ is a monotonic mapping from candidates to candidates, hence it admits a least fixed point in the lattice of candidates. Next, we check that monotonicity is preserved. Let us consider $\mathcal{E}$ and $\mathcal{E}'$ differing only on a variable p occurring only positively in $\mu B\vec{t}$, with $\mathcal{E}(p) \subseteq \mathcal{E}'(p)$. Unfolding the definition, we have $|\mu B\vec{t}|^{\mathcal{E}} = \mathit{lfp}(\phi)$ and $|\mu B\vec{t}|^{\mathcal{E}'} = \mathit{lfp}(\phi')$. By induction hypothesis, $\phi(\mathcal{X}) \subseteq \phi'(\mathcal{X})$ for any candidate $\mathcal{X}$, and in particular $\phi(|\mu B\vec{t}|^{\mathcal{E}'}) \subseteq \phi'(|\mu B\vec{t}|^{\mathcal{E}'}) = |\mu B\vec{t}|^{\mathcal{E}'}$. The least fixed point being contained in all prefixed points, we obtain the expected result: $|\mu B\vec{t}|^{\mathcal{E}} = \mathit{lfp}(\phi) \subseteq |\mu B\vec{t}|^{\mathcal{E}'}$. Antimonotonicity is established in a symmetric fashion.

Notation 1 (Interpretation of predicates and operators)**.** *When P is a predicate, i.e., an object of type* $\vec{\gamma} \to o$*, we define* $|P|^{\mathcal{E}}$ *to be the mapping* $\vec{t} \mapsto |P\vec{t}|^{\mathcal{E}}$*. For a predicate operator B, i.e., an object of type* $(\vec{\gamma} \to o) \to o$*, we define* $|B|^{\mathcal{E}}$ *to be the mapping* $\mathcal{X} \mapsto |Bp|^{\mathcal{E}+\langle p,\mathcal{X}\rangle}$*. For conciseness we write directly* $|B\mathcal{X}\vec{t}|^{\mathcal{E}}$ *for* $|\lambda p.\ Bp\vec{t}|^{\mathcal{E}}\mathcal{X}$*, which is also equivalent to* $|B|^{\mathcal{E}}\mathcal{X}\vec{t}$*.*

Lemma 4. *Interpretation commutes with second-order substitution:* $|B[P/p]|^{\mathcal{E}} = |B|^{\mathcal{E}+\langle p,|P|^{\mathcal{E}}\rangle}$*.*

Proof: Straightforward, by induction on B. ∎

We naturally extend the interpretation to typing contexts: if $\Gamma = (\alpha_1 : P_1, \ldots, \alpha_n : P_n)$, $|\Gamma|^{\mathcal{E}} = (\alpha_1 : |P_1|^{\mathcal{E}}, \ldots, \alpha_n : |P_n|^{\mathcal{E}})$. We also write $\sigma \in |\Gamma|^{\mathcal{E}}$ when σ is of the form $[\pi_1/\alpha_1, \ldots, \pi_n/\alpha_n]$ with $\pi_i \in |P_i|^{\mathcal{E}}$ for all i.

Definition 10. *If* π *is a proof term with free variables* $\alpha_1, \ldots, \alpha_n$ *and* $\mathcal{Y}, \mathcal{X}_1, \ldots, \mathcal{X}_n$ *are reducibility candidates, we say that* π *is* $(\alpha_1 : \mathcal{X}_1, \ldots, \alpha_n : \mathcal{X}_n \vdash \mathcal{Y})$-reducible *if* $\pi[\pi'_i/\alpha_i]_i \in \mathcal{Y}$ *for any* $(\pi'_i)_i \in (\mathcal{X}_i)_i$*. When it is not ambiguous, we may omit the variables and simply say that* π *is* $(\mathcal{X}_1, \ldots, \mathcal{X}_n \vdash \mathcal{Y})$*-reducible.*

Definition 11. *A pre-model* $\mathcal{M}$ *is a* pre-model of $\equiv$ *iff any two congruent formulas have the same interpretation with respect to* $\mathcal{M}$*.*

In the rest of this section, we assume a pre-model of the congruence, and we shall establish that any term $\Gamma \vdash \pi : P$ is $(|\Gamma|^{\mathcal{E}} \vdash |P|^{\mathcal{E}})$-reducible. In order to do so, we prove *adequacy* lemmas, showing that each typing rule can be simulated in the interpretation.

Lemma 5. *The following adequacy properties hold.*

- ($\supset$)
  - *If* π *is* $(\alpha : |P|^{\mathcal{E}} \vdash |Q|^{\mathcal{E}})$*-reducible, then* $\lambda\alpha.\pi \in |P \supset Q|^{\mathcal{E}}$*.*
  - *If* $\pi \in |P \supset Q|^{\mathcal{E}}$ *and* $\pi' \in |P|^{\mathcal{E}}$*, then* $\pi\pi' \in |Q|^{\mathcal{E}}$*.*
- ($\wedge$)
  - *If* $\pi_1 \in |P_1|^{\mathcal{E}}$ *and* $\pi_2 \in |P_2|^{\mathcal{E}}$*, then* $\langle\pi_1, \pi_2\rangle \in |P_1 \wedge P_2|^{\mathcal{E}}$*.*
  - *If* $\pi \in |P_1 \wedge P_2|^{\mathcal{E}}$*, then* $\mathrm{proj}_1(\pi) \in |P_1|^{\mathcal{E}}$ *and* $\mathrm{proj}_2(\pi) \in |P_2|^{\mathcal{E}}$*.*
- ($\vee$)
  - *If* $\pi \in |P_i|^{\mathcal{E}}$ *for* $i \in \{1, 2\}$*, then* $\mathrm{in}_i(\pi) \in |P_1 \vee P_2|^{\mathcal{E}}$*.*
  - *If* $\pi \in |P_1 \vee P_2|^{\mathcal{E}}$ *and each* π_i *is* $(\alpha : |P_i|^{\mathcal{E}} \vdash |Q|^{\mathcal{E}})$*-reducible, then* $\delta_\vee(\pi, \alpha.\pi_1, \alpha.\pi_2) \in |Q|^{\mathcal{E}}$*.*
- ($\top$)
  - *The proof* $\langle\rangle$ *belongs to* $|\top|^{\mathcal{E}}$*.*
- ($\bot$)
  - *If* $\pi \in |\bot|^{\mathcal{E}}$*, then* $\delta_\bot(\pi) \in |P|^{\mathcal{E}}$ *for any P.*
- ($\forall$)
  - *If* $\pi[t/x] \in |P[t/x]|^{\mathcal{E}}$ *for any t, then* $\lambda x.\pi \in |\forall x.\ P|^{\mathcal{E}}$*.*
  - *If* $\pi \in |\forall x.\ P|^{\mathcal{E}}$*, then* $\pi t \in |P[t/x]|^{\mathcal{E}}$*.*
- ($\exists$)
  - *If* $\pi \in |P[t/x]|^{\mathcal{E}}$*, then* $\langle t, \pi\rangle \in |\exists x.\ P|^{\mathcal{E}}$*.*
  - *If* $\pi \in |\exists x.\ P|^{\mathcal{E}}$ *and* $\pi'[t/x]$ *is* $(\alpha : |P[t/x]|^{\mathcal{E}} \vdash |Q|^{\mathcal{E}})$*-reducible for any t, then* $\delta_\exists(\pi, x.\alpha.\pi') \in |Q|^{\mathcal{E}}$*.*
- ($=$)
  - *It is always the case that* $\mathrm{refl} \in |t = t|^{\mathcal{E}}$*.*
  - *If* $\pi \in |t\theta = t'\theta|^{\mathcal{E}}$*,* $\sigma \in |\Gamma\theta|^{\mathcal{E}}$ *and* $\pi'_i\theta'$ *is* $(|\Gamma\theta_i\theta'|^{\mathcal{E}} \vdash |P\theta_i\theta'|^{\mathcal{E}})$*-reducible for any i and* θ'*, then* $\delta_=(\Gamma, \theta, \sigma, t, t', P, \pi, (\theta_i.\pi_i)_i) \in |P\theta|^{\mathcal{E}}$*.*
- (μ)
  - *If* $\pi \in |B(\mu B)\vec{t}|^{\mathcal{E}}$*, then* $\mu(B, \vec{t}, \pi) \in |\mu B\vec{t}|^{\mathcal{E}}$*.*
- (ν)
  - *If* $\pi \in |\nu B\vec{t}|^{\mathcal{E}}$*, then* $\delta_\nu(B, \vec{t}, \pi) \in |B(\nu B)\vec{t}|^{\mathcal{E}}$*.*

Proof: We illustrate this standard proof technique on a few cases; more details may be found in Appendix A. All introduction cases follow a similar pattern. Consider for example the case of least fixed-points. We need to show that for any $\pi \in |B(\mu B)\vec{t}|^{\mathcal{E}}$, $\mu(B, \vec{t}, \pi) \in |\mu B\vec{t}|^{\mathcal{E}} = \mathit{lfp}(\phi)(\vec{t}) = \phi(|\mu B|^{\mathcal{E}})(\vec{t}) = \{\ \rho \in \mathcal{SN} \mid \rho \to^* \mu(B, \vec{t}, \pi')$ implies $\pi' \in |B(\mu B)\vec{t}|^{\mathcal{E}}\ \}$. Indeed, for any reduction $\mu(B, \vec{t}, \pi) \to^* \mu(B, \vec{t}, \pi')$ it must be the case that $\pi \to^* \pi'$ and thus $\pi' \in |B(\mu B)\vec{t}|^{\mathcal{E}}$.

The elimination of greatest fixed points follows immediately from the definition of the interpretation since $|\nu B\vec{t}|^{\mathcal{E}} = \{\ \pi \mid \delta_\nu(B, \vec{t}, \pi) \in |B(\nu B)\vec{t}|^{\mathcal{E}}\ \}$. Other elimination cases follow a scheme that we illustrate on equality elimination. Under the above mentionned conditions, we show that $\delta_=(\Gamma, \theta, \sigma, t, t', P, \pi, (\theta_i.\pi_i)_i) \in |P\theta|^{\mathcal{E}}$. We proceed by induction on the strong normalizability of the subderivations π, σ and π_i. In order to show that a neutral term belongs to a candidate, it suffices to consider all its one-step reducts. Reductions occurring inside subterms are handled by induction hypothesis. We may also have a toplevel redex when $t\theta \equiv t'\theta$ and $\pi = \mathit{refl}$, reducing to $\pi_i\theta'\sigma$ where θ' is such that $\theta_i\theta' \equiv \theta$. By hypothesis, $\pi_i\theta'$ is $(|\Gamma\theta_i\theta'|^{\mathcal{E}} \vdash |P\theta_i\theta'|^{\mathcal{E}})$-reducible and $\sigma \in |\Gamma\theta|^{\mathcal{E}} = |\Gamma\theta_i\theta'|^{\mathcal{E}}$, and thus we have $\pi_i\theta'\sigma \in |P\theta|^{\mathcal{E}}$ as expected. ∎

Although adequacy is easily proved for our new equality formulation, a few important observations should be made here. First, the proof crucially relies on the fact that we are considering only syntactic pre-models, and not the general notion of pre-model of Dowek and Werner where terms may be interpreted in arbitrary structures. This requirement makes sense conceptually, since closed-world equality internalizes the fact that equality can only hold when the congruence

$$
\begin{array}{rcl}
|\bot|^{\mathcal{E}} = |\top|^{\mathcal{E}} = |u = v|^{\mathcal{E}} = \mathcal{SN} & & |p\ t_1 \dots t_n|^{\mathcal{E}} = \mathcal{E}(p)(t_1, \dots, t_n) \qquad |a\ t_1 \dots t_n|^{\mathcal{E}} = \hat{a}(t_1, \dots, t_n) \\
|P \supset Q|^{\mathcal{E}} & = & \{ \pi \in \mathcal{SN} \mid \pi \to^* \lambda\alpha.\pi_1 \text{ implies } \pi_1[\pi'/\alpha] \in |Q|^{\mathcal{E}} \text{ for any } \pi' \in |P|^{\mathcal{E}} \} \\
|P \wedge Q|^{\mathcal{E}} & = & \{ \pi \in \mathcal{SN} \mid \pi \to^* \langle \pi_1, \pi_2 \rangle \text{ implies } \pi_1 \in |P|^{\mathcal{E}} \text{ and } \pi_2 \in |Q|^{\mathcal{E}} \} \\
|P_1 \vee P_2|^{\mathcal{E}} & = & \{ \pi \in \mathcal{SN} \mid \pi \to^* in_i(\pi') \text{ implies } \pi' \in |P_i|^{\mathcal{E}} \} \\
|\forall x.\ P|^{\mathcal{E}} & = & \{ \pi \in \mathcal{SN} \mid \pi \to^* \lambda x.\pi' \text{ implies } \pi'[t/x] \in |P[t/x]|^{\mathcal{E}} \text{ for any } t \} \\
|\exists x.\ P|^{\mathcal{E}} & = & \{ \pi \in \mathcal{SN} \mid \pi \to^* \langle t, \pi' \rangle \text{ implies } \pi'[t/x] \in |P[t/x]|^{\mathcal{E}} \} \\
|\mu B \vec{t}|^{\mathcal{E}} & = & lfp(\phi)(\vec{t}) \text{ where } \phi(\mathcal{X}) = \vec{t} \mapsto \{ \pi \in \mathcal{SN} \mid \pi \to^* \mu(B, \vec{t}, \pi') \text{ implies } \pi' \in |Bp\vec{t}|^{\mathcal{E}+\langle p, \mathcal{X} \rangle} \} \\
|\nu B \vec{t}|^{\mathcal{E}} & = & gfp(\phi)(\vec{t}) \text{ where } \phi(\mathcal{X}) = \vec{t} \mapsto \{ \pi \mid \delta_\nu(B, \vec{t}, \pi) \in |Bp\vec{t}|^{\mathcal{E}+\langle p, \mathcal{X} \rangle} \}
\end{array}
$$

Fig. 6: Interpretation of formulas as candidates

allows it, and is thus incompatible with further equalities that could hold in non-trivial semantic interpretations. Second, the suspension of proof-level substitutions in equality elimination goes hand in hand with the independence of interpretations for different predicate instances, which in turn is necessary to interpret recursive definitions. Indeed, when applying a proof-level substitution $\sigma \in |\Gamma|^{\mathcal{E}}$ on an eager equality elimination, we are forced us to apply the *csu* substitutions on σ, and we need $\sigma \in |\Gamma\theta_i|^{\mathcal{E}}$ which essentially forces us to have a term-independent interpretation [9].

We now address the adequacy of functoriality, induction and coinduction.

Lemma 6. *Let π be a proof, and let $\mathcal{X}$ and $\mathcal{X}'$ be predicate candidates such that $\pi[\vec{t}/\vec{x}]$ is $(\alpha : \mathcal{X}\vec{t} \vdash \mathcal{X}'\vec{t})$-reducible for any $\vec{t}$. If B is a monotonic (resp. antimonotonic) operator, then $F^+_B(\vec{x}.\alpha.\pi) \in |B\mathcal{X} \supset B\mathcal{X}'|$ (resp. $F^-_B(\vec{x}.\alpha.\pi) \in |B\mathcal{X}' \supset B\mathcal{X}|$).*

Lemma 7. *Let π be a proof and $\mathcal{X}$ a predicate candidate. If $\pi[\vec{t}/\vec{x}]$ is $(\alpha : |B|\mathcal{X}\vec{t} \vdash \mathcal{X}\vec{t})$-reducible for any $\vec{t}$, then $\delta_\mu(\beta, \vec{x}.\alpha.\pi)$ is $(\beta : |\mu B\vec{t}| \vdash \mathcal{X}\vec{t})$-reducible for any $\vec{t}$.*

Lemma 8. *Let π be a proof and $\mathcal{X}$ a predicate candidate. If $\pi[\vec{t}/\vec{x}]$ is $(\alpha : \mathcal{X}\vec{t} \vdash |B|\mathcal{X}\vec{t})$-reducible for any $\vec{t}$, then $\nu(\beta, \vec{x}.\alpha.\pi)$ is $(\beta : \mathcal{X}\vec{t} \vdash |\nu B\vec{t}|)$-reducible for any $\vec{t}$.*

Proof: Those lemmas must be proved simultaneously, in a generalized form that is detailed in the appendix. There is no essential difficulty in proving the functoriality lemma, using previously proved adequacy properties as well as the other two lemmas for the fixed point cases. The next two lemmas are the interesting ones, since they involve using the properties of the fixed point interpretations to justify the (co)induction rules. In the case of induction, we need to establish that $\delta_\mu(\rho, \vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t}$ when $\rho \in |\mu B\vec{t}|$. In order to do this, it suffices to show that $\mathcal{Y} := \vec{t} \mapsto \{ \rho \mid \delta_\mu(\rho, \vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t} \}$ is included in $|\mu B|$. This follows from the fact that $\mathcal{Y}$ is a pre-fixed point of the operator ϕ such that $|\mu B| = lfp(\phi)$, which can be proved easily using the adequacy property for functoriality. We proceed similarly for the coinduction rule, showing that

$$
\begin{array}{l}
\mathcal{Y} := \vec{t} \mapsto \{ \pi \in \mathcal{SN} \mid \pi \to^* \nu(\rho, \vec{x}.\alpha.\pi) \text{ implies } \rho \in \mathcal{X}\vec{t} \text{ and} \\
\quad \pi[\vec{t}/\vec{x}] \text{ is } (\alpha : \mathcal{X}\vec{t} \vdash |B|\mathcal{X}\vec{t})\text{-reducible for any } \vec{t} \}
\end{array}
$$

is a post-fixed point of the operator ϕ such that $|\nu B| = gfp(\phi)$. In both cases, note that the candidate $\mathcal{Y}$ is *a priori* not the interpretation of any predicate; this is where we use the power of reducibility candidates. ■

Theorem 2 (Adequacy)**.** *Let $\equiv$ be a congruence, $\mathcal{M}$ a pre-model of $\equiv$ and $\Gamma \vdash \pi : P$ a derivable judgment. Then $\pi\sigma \in |P|$ for any substitution $\sigma \in |\Gamma|$.*

Proof: By induction on the height of π, using the previous adequacy properties. ■

The usual corollaries hold. Since variables belong to any candidate by condition (3), we can take σ to be the identity substitution, and obtain that any well-typed proof is strongly normalizable. Together with Lemma 3, this means that our logic is consistent. Note that the suspended computations in the (co)induction and equality elimination rules do not affect these corollaries, because they can only occur in normal forms of specific types. For instance, equality elimination cannot hide a non-terminating computation if there is no equality assumption in the environment.

V. Recursive definitions

We now identify a class of rewrite rules relative to which we can always build a pre-model. This class supports recursive definitions whose use we illustrate through a sound formalization of a Tait-style argument.

A. Recursive rewriting that admits a pre-model

The essential idea behind recursive definitions is that they are formed gradually, following the inductive structure of one of their arguments, or more generally a well-founded order on arguments. In order to reflect this idea into a pre-model construction, we need to identify all the atom interpretations that could be involved in the interpretation of a given formula. This is the purpose of the next definition.

Definition 12. *We say that P* may occur *in Q when $P = P'\theta$, P' occurs in Q, and θ is a substitution for variables quantified over in Q.*

For example, $(a\,t)$ may occur in $(a'\ x \wedge \exists y.\ a\,y)$ for any t.

Theorem 3. *Let $\equiv$ be a congruence defined by a rewrite system rewriting terms to terms and atomic propositions to propositions, and let $\mathcal{M}$ be a pre-model of $\equiv$. Consider the addition of new predicate symbols $a_1, \dots, a_n$ in the language, together with the extension of the congruence resulting from the addition of rewrite rules of the form $a_i\vec{t} \rightsquigarrow B$. There* *is a pre-model of the extended congruence in the extended language, provided that the following conditions hold.*

(1) *If* $(a_i\vec{t})\theta \equiv (a_i\vec{t'})\theta'$, $a_i\vec{t} \rightsquigarrow B$ *and* $a_i\vec{t'} \rightsquigarrow B'$, *then* $B\theta \equiv B'\theta'$.

(2) *There exists a well-founded order* $\prec$ *such that* $a_j\vec{t'} \prec (a_i\vec{t})\theta$ *whenever* $a_i\vec{t} \rightsquigarrow B$ *and* $a_j\vec{t'}$ *may occur in* $B'\theta$.

Note that condition (1) is not obviously satisfied, even when there is a single rule per atom. Consider, for example, $a\,(0 \times x) \rightsquigarrow a'\,x$ in a setting where $0 \times x \equiv 0$: our condition requires that $a'\,x \equiv a'\,y$ for any x and y, which is *a priori* not guaranteed. Condition (2) restricts the use of quantifiers but still allows useful constructions. Consider for example the Ackermann relation, built using a double induction on its first two parameters: $ack\ 0\ x\ (s\ x) \rightsquigarrow \top$, $ack\ (s\ x)\ 0\ y \rightsquigarrow ack\ x\ (s\ 0)\ y$ and $ack\ (s\ x)\ (s\ y)\ z \rightsquigarrow \exists r.\ ack\ (s\ x)\ y\ r \wedge ack\ x\ r\ z$. The third rule requires that $ack\ x\ r\ z \prec ack\ (s\ x)\ (s\ y)\ z$ for any x, y, z and r, which is indeed satisfied with a lexicographic ordering.

Proof: We only present the main idea here; a detailed proof may be found in the appendix. We first build pre-models $\mathcal{M}^{a_i\vec{t}}$ that are compatible with instances $a_j\vec{t'} \rightsquigarrow B$ of the new rewrite rules for $a_j\vec{t'} \leq a_i\vec{t}$. This is done gradually following the order $\prec$, using a well-founded induction on $a_i\vec{t}$. We build $\mathcal{M}^{a_i\vec{t}}$ by aggregating smaller pre-models $\mathcal{M}^{a_j\vec{t'}}$ for $a_j\vec{t'} \prec a_i\vec{t}$, and adding the interpretation $\hat{a}_i\vec{t}$. To define it, we consider rule instances of the form $a_i\vec{t} \rightsquigarrow B$. If there is none we use a dummy interpretation: $\hat{a}_i\vec{t} = \mathcal{SN}$. Otherwise, condition (1) imposes that there is essentially a single possible such rewrite modulo the congruence, so it suffices to choose $|B|$ as the interpretation $\hat{a}_i\vec{t}$ to satisfy the new rewrite rules. Finally, we aggregate interpretations from all the pre-models $\mathcal{M}^{a_i\vec{t}}$ to obtain a pre-model of the full extended congruence. ■

This result can be used to obtain pre-models for complex definition schemes, such as ones that iterate and interleave groups of fixed-point and recursive definitions. Consider, for example, $a\ (s\ n) \rightsquigarrow a\ n \supset a\ (s\ n)$. While this rewrite rule does not directly satisfy the conditions of Theorem 3, it can be rewritten into the form $a\ (s\ n) \rightsquigarrow \mu Q.\ a\ n \supset Q$, which does satisfy these conditions.

B. An application of recursive definitions

Our example application is the formalization of the Tait-style argument of strong normalizability for the simply typed λ-calculus. We assume term-level sorts *tm* and *ty* corresponding to representations of λ-terms and simple types, and symbols $\iota : ty$, $arrow : ty \to ty \to ty$, $app : tm \to tm \to tm$ and $abs : (tm \to tm) \to tm$. We identify well-formed types through an inductive predicate:

$$isty \stackrel{def}{=} \mu(\lambda T \lambda t.\ t = \iota \vee \exists t' \exists t''.\ t = arrow\ t'\ t'' \wedge T\ t' \wedge T\ t'')$$

We assume a definition of term reduction and strong normalization, denoting the latter predicate by *sn*. Finally, we define *red m t*, expressing that *m* is a reducible λ-term of type *t*, by the following rewrite rules:

$$\begin{array}{rcl} red\ m\ \iota & \rightsquigarrow & sn\ m \\ red\ m\ (arrow\ t\ t') & \rightsquigarrow & \forall n.\ red\ n\ t \supset red\ (app\ m\ n)\ t' \end{array}$$

This definition satisfies the conditions of Theorem 3, taking as $\prec$ the order induced by the subterm ordering on the second argument of *red*. We can thus safely use it.

With these definitions, our logic allows us to mirror very closely the strong normalization proof presented in [16]. For instance, consider proving that reducible terms are strongly normalizing:

$$\forall m \forall t.\ isty\ t \supset red\ m\ t \supset sn\ m$$

The paper proof is by induction on types, which corresponds in the formalization to an elimination on *isty t*. In the base case, we have to derive $red\ m\ \iota \supset sn\ m$ which is simply an instance of $P \supset P$ modulo our congruence. In the arrow case, we must prove $red\ m\ (arrow\ t\ t') \supset sn\ m$. The hypothesis $red\ m\ (arrow\ t\ t')$ is congruent to $\forall n.\ red\ n\ t \supset red\ (app\ m\ n)\ t'$ and we can show that variables are always reducible,[1] which gives us $red\ (app\ m\ x)\ t'$. From there, we obtain $sn\ (app\ m\ x)$ by induction hypothesis, from which we can deduce $sn\ m$ with a little more work.

The full formalization, which is too detailed to present here, is available from the authors. This formalization has been tested using the proof assistant Abella [20]. The logic that underlies Abella features fixed-point definitions, closed-world equality and generic quantification. The last notion is useful when dealing with binding structures, and we have employed it in our formalization although it is not available yet in our logic. Abella does not actually support recursive definitions. To get around this fact, we have entered the one we need as an inductive definition, and ignored the warning provided about the non-monotonic clause while making sure to use an unfolding of this inductive definition in the proof only when this is allowed for recursive definitions. In the future, we plan to extend Abella to support recursive definitions based on the theory developed in this paper. This would mean allowing such definitions as a separate class, building in a test that they satisfy the criterion described in Theorem 3 and properly restricting the use of these definitions in proofs. Such an extension is obviously compatible with all the current capabilities of Abella and would support additional reasoning that is justifiably sound.

VI. Related and Future Work

The logical system that we have developed is obviously related to deduction modulo. In essence, it extends that system with a simple yet powerful treatment of fixed-point definitions. The additional power is obtained from two new features: fixed-point combinators and closed-world equality. If our focus is only on provability, the capabilities arising from these features may perhaps be encoded in deduction modulo. Dowek and Werner provide an encoding of arithmetic in deduction modulo, and also show how to build pre-models for some more general fixed-point constructs [18]. Regarding equality, Allali [21] has shown that a more algorithmic version

[1] This actually has to be proved simultaneously with $red\ m\ t \supset sn\ m$, but we ignore it for the simplicity of the presentation.

of equality may be defined through the congruence, which allows to simplify some equations by computing. Thus, it simulates some aspects of closed-world equality. However, the principle of substitutivity has to be recovered through a complex encoding involving inductions on the term structures. In any case, our concern here is not simply with provability; in general, we do not follow the project of deduction modulo to have a logic as basic as possible in which stronger systems are then encoded. Rather, we seek to obtain meaningful proof structures, whose study can reveal useful information. For instance, in the context of proof-search, it has been shown that a direct treatment of fixed-point definitions allows for stronger focused proof systems [9] which have served as a basis for several proof-search implementations [10], [11]. This goal also justifies why we do not simply use powerful systems such as the Calculus of Inductive Constructions [3] which obviously supports inductive as well as recursive definitions; here again we highlight the simplicity of our (co)induction rules and of our rich equality elimination principle.

Our logic is also related to logics of fixed-point definitions [4]–[6]. The system we have described represents an advance over these logics in that it adds to them a rewriting capability. As we have seen, this capability can be used to support recursive definitions as well as to blend computation and deduction in natural ways. This paper also makes important contributions to the understanding of closed-world equality. We have shown that it is compatible with an equational theory on terms. We have, in addition, resolved some problematic issues related to this notion that affect the stability of finite proofs under reduction. This has allowed us to prove for the first time a strong normalizability result for logics of fixed-point definitions. Our calculus is, at this stage, missing a treatment of generic quantification present in some of the alternative logics [12]–[14]. We plan to include this feature in the future, and do not foresee any difficulty in doing so since it has typically been added in a modular fashion to such logics. This addition would make our logic an excellent choice for formalizing the meta-theory of computational and logical systems.

An important topic for further investigation of our system is proof search. The distinction between computation and deduction is critical for theorem proving with fixed point definitions. For instance, in the Tac system [11], which is based on logics of definitions, automated (co)inductive theorem proving relies heavily on ad-hoc annotations that identify computations. In that context, our treatment of recursive definitions seems like a good candidate more a more principled separation of computation and deduction. Finally, now that we have refactored equality rules to simplify the proof normalization process, we should study their proof search behavior. The new equality elimination rule seems difficult to analyze at first. However, we hope to gain some insights from studying its use in settings where the old rule (which it subsumes) is practically satisfactory, progressively moving to newer contexts where it offers advantages. We note in this regard that the new complexity is in fact welcome: the earlier infinitely branching treatments of closed-world equality had a simple proof-search treatment in theory, but did not provide a useful handle to study the practical difficulties of automated theorem proving with complex equalities.

References

[1] N. P. Mendler, "Inductive types and type constraints in the second order lambda calculus," *Annals of Pure and Applied Logic*, vol. 51, no. 1, pp. 159–172, 1991.

[2] R. Matthes, "Monotone fixed-point types and strong normalization," in *CSL 1998: Computer Science Logic*, ser. LNCS, G. Gottlob, E. Grandjean, and K. Seyr, Eds., Berlin, 1999, vol. 1584, pp. 298–312.

[3] C. Paulin-Mohring, "Inductive definitions in the system Coq: Rules and properties," in *Proceedings of the International Conference on Typed Lambda Calculi and Applications*, M. Bezem and J. F. Groote, Eds. Utrecht, The Netherlands: Springer LNCS 664, Mar. 1993, pp. 328–345.

[4] P. Schroeder-Heister, "Rules of definitional reflection," in *8th Symp. on Logic in Computer Science*, M. Vardi, Ed., IEEE Computer Society Press. IEEE, Jun. 1993, pp. 222–232.

[5] R. McDowell and D. Miller, "Cut-elimination for a logic with definitions and induction," *Theoretical Computer Science*, vol. 232, pp. 91–119, 2000.

[6] A. Tiu and A. Momigliano, "Induction and co-induction in sequent calculus," *CoRR*, vol. abs/0812.4727, 2008.

[7] A. Tiu, "A logical framework for reasoning about logical specifications," Ph.D. dissertation, Pennsylvania State University, May 2004.

[8] D. Baelde, "A linear approach to the proof-theory of least and greatest fixed points," Ph.D. dissertation, Ecole Polytechnique, Dec. 2008.

[9] ——, "Least and greatest fixed points in linear logic," vol. 13, no. 1, Jan. 2012, ACM Transactions on Computational Logic.

[10] D. Baelde, A. Gacek, D. Miller, G. Nadathur, and A. Tiu, "The Bedwyr system for model checking over syntactic expressions," in *21th Conf. on Automated Deduction (CADE)*, ser. LNAI, F. Pfenning, Ed., no. 4603. New York: Springer, 2007, pp. 391–397.

[11] D. Baelde, D. Miller, and Z. Snow, "Focused inductive theorem proving," in *Fifth International Joint Conference on Automated Reasoning*, ser. LNCS, J. Giesl and R. Hähnle, Eds., no. 6173, 2010, pp. 278–292.

[12] A. Gacek, "A framework for specifying, prototyping, and reasoning about computational systems," Ph.D. dissertation, University of Minnesota, 2009.

[13] A. Gacek, D. Miller, and G. Nadathur, "Nominal abstraction," *Information and Computation*, vol. 209, no. 1, pp. 48–73, 2011.

[14] D. Miller and A. Tiu, "A proof theory for generic judgments," *ACM Trans. on Computational Logic*, vol. 6, no. 4, pp. 749–783, Oct. 2005.

[15] W. W. Tait, "Intensional interpretations of functionals of finite type I," *J. of Symbolic Logic*, vol. 32, no. 2, pp. 198–212, 1967.

[16] J.-Y. Girard, P. Taylor, and Y. Lafont, *Proofs and Types*. Cambridge University Press, 1989.

[17] G. Dowek, T. Hardin, and C. Kirchner, "Theorem proving modulo," *J. of Automated Reasoning*, vol. 31, no. 1, pp. 31–72, 2003.

[18] G. Dowek and B. Werner, "Proof normalization modulo," *Journal of Symbolic Logic*, vol. 68, no. 4, pp. 1289–1316, 2003.

[19] G. Burel, "Unbounded proof-length speed-up in deduction modulo," in *CSL 2007: Computer Science Logic*, ser. LNCS, J. Duparc and T. A. Henzinger, Eds., vol. 4646. Springer, 2007, pp. 496–511.

[20] A. Gacek, "The Abella interactive theorem prover (system description)," in *Fourth International Joint Conference on Automated Reasoning*, ser. LNCS, A. Armando, P. Baumgartner, and G. Dowek, Eds., vol. 5195. Springer, 2008, pp. 154–161.

[21] L. Allali, "Algorithmic equality in heyting arithmetic modulo," in *TYPES*, ser. LNCS, M. Miculan, I. Scagnetto, and F. Honsell, Eds., vol. 4941. Springer, 2007, pp. 1–17.

Appendix

A. Proof of Lemma 5

All introduction rules are treated in a similar fashion:

- *If* π *is* $(\alpha : |P| \vdash |Q|)$*-reducible, then* $\lambda\alpha.\pi \in |P \supset Q|$.
  First, $\lambda\alpha.\pi$ is SN, like all reducible proof-terms, because variables belong to all candidates, and candidates are sets of SN proofs. Now, assuming $\lambda\alpha.\pi \to^* \lambda\alpha.\pi'$, we seek to establish that $\pi'[\pi''/\alpha] \in |Q|$ for any $\pi'' \in |P|$. By definition of reducibility, $\pi[\pi''/\alpha]$ belongs to $|Q|$, and we conclude by stability of candidates under reduction since $\pi[\pi''/\alpha] \to^* \pi'[\pi''/\alpha]$.
- The cases for $\wedge$, $\vee$ and $\exists$ are proved similarly.
- The cases for $\top$ and equality are trivial.
- *If* $\pi[t/x] \in |P[t/x]|$ *for any* t*, then* $\lambda x.\pi \in |\forall x.\ P|$.
  Assume $\lambda x.\pi \to^* \lambda x.\pi'$. It must be the case that $\pi \to^* \pi'$, and for any $\vec{t}$ we have $\pi[t/x] \to^* \pi'[t/x]$ by Proposition 1 and thus $\pi'[t/x] \in |P[t/x]|$ as needed.
- *If* $\pi \in |B(\mu B)\vec{t}|$*, then* $\mu(B, \vec{t}, \pi) \in |\mu B\vec{t}|$.
  We need to show that $\mu(B,\vec{t},\pi) \in |\mu B\vec{t}| = lfp(\phi)(\vec{t}) = \phi(|\mu B|)(\vec{t}) = \{\ \rho \in \mathcal{SN} \mid \rho \to^* \mu(B,\vec{t},\pi')$ implies $\pi' \in |B(\mu B)\vec{t}|\ \}$. Indeed, for any reduction $\mu(B,\vec{t},\pi) \to^* \mu(B,\vec{t},\pi')$ it must be the case that $\pi \to^* \pi'$ and thus $\pi' \in |B(\mu B)\vec{t}|$.

Elimination rules also follow a common scheme:

- *If* $\pi \in |P \supset Q|$ *and* $\pi' \in |P|$*, then* $\pi\,\pi' \in |Q|$.
  We proceed by induction on the strong normalizability of π and π'. By the candidate of reducibility condition on neutral terms, it suffices to show that all immediate reducts $\pi\,\pi' \to \pi''$ belong to $|Q|$. If π'' is obtained by a reduction inside π or π', then we conclude by induction hypothesis since the resulting subterm still belongs to the expected interpretation. Otherwise, it must be that $\pi = \lambda\alpha.\rho$ and the reduct is $\rho[\pi'/\alpha]$. In that case we conclude by definition of $\pi \in |P \supset Q|$.
- The cases of $\wedge$, $\vee$ and $\bot$ are treated similarly.
- *If* $\pi \in |\forall x.\ P|$*, then* $\pi\,t \in |P[t/x]|$.
  We proceed by induction on the strong normalizability of π, considering all one-step reducts of the neutral term $\pi\,t$. Internal reductions are handled by induction hypothesis. If $\pi = \lambda x.\pi'$, our term may reduce at toplevel into $\pi'[t/x]$. In that case we conclude by definition of $|\forall x.\ P|$.
- *If* $\pi \in |\exists x.\ P|$ *and* $\pi'[t/x]$ *is* $(\alpha : |P[t/x]| \vdash |Q|)$*-reducible for any* t*, then* $\delta_\exists(\pi, x.\alpha.\pi') \in |Q|$.
  We proceed by induction on the strong normalizability of π and π', considering all one-step reducts. The internal reductions are handled by induction hypothesis. A toplevel reduction into $\pi'[t/x][\pi''/\alpha]$ may occur when $\pi = \langle t, \pi''\rangle$ in which case we have $\pi'' \in |P[t/x]|$ by hypothesis on π and definition of $|\exists x.\ P|$. We conclude by hypothesis on $\pi'[t/x]$.
- *If* $\pi \in |t\theta = t'\theta|$*,* $\sigma \in |\Gamma\theta|$ *and* $\pi_i'\theta'$ *is* $(|\Gamma\theta_i\theta'| \vdash |P\theta_i\theta'|)$*-reducible for any* i *and* θ'*, then* $\delta_=(\Gamma, \theta, \sigma, t, t', P, \pi, (\theta_i.\pi_i)_i) \in |P\theta|$.
  We proceed by induction on the strong normalizability of the subderivations π, σ and π_i. In order to show that a neutral term belongs to a candidate, it suffices to consider all its one-step reducts. Reductions occurring inside subterms are handled by induction hypothesis. We may also have a toplevel redex when $t\theta \equiv t'\theta$ and $\pi = refl$, reducing to $\pi_i\theta'\sigma$ where θ' is such that $\theta_i\theta' \equiv \theta$. By hypothesis, $\pi_i\theta'$ is $(|\Gamma\theta_i\theta'| \vdash |P\theta_i\theta'|)$-reducible and $\sigma \in |\Gamma\theta| = |\Gamma\theta_i\theta'|$, and thus we have $\pi_i\theta'\sigma \in |P\theta|$ as expected.
- The case of δ_ν is singular: it follows directly from the definition of the interpretation of greatest fixed points.

B. Proof of Lemmas 6, 7 and 8

Let us first introduce the following notation for conciseness: we say that π is $(\vec{x}, \mathcal{X}\vec{x} \vdash \mathcal{Y}\vec{x})$-reducible when $\pi[\vec{t}/\vec{x}]$ is $(\mathcal{X}\vec{t} \vdash \mathcal{Y}\vec{t})$-reducible for any $\vec{t}$.

We prove the three lemmas simultaneously, generalized as follows for a predicate operator B of second-order arity[2] $n+1$, predicates $\vec{A}$ and predicate candidates $\vec{Z}$:

(1) For any $(\vec{x}, \mathcal{X}\vec{x} \vdash \mathcal{X}'\vec{x})$-reducible π, $F^+_{B\vec{A}}(\vec{x}.\alpha.\pi) \in |B\vec{Z}\mathcal{X} \supset B\vec{Z}\mathcal{X}'|$.

(2) For any $(\vec{x}, \mathcal{X}\vec{x} \vdash \mathcal{X}'\vec{x})$-reducible π, $F^-_{B\vec{A}}(\vec{x}.\alpha.\pi) \in |B\vec{Z}\mathcal{X}' \supset B\vec{Z}\mathcal{X}|$.

(3) For any $(\vec{x}, |B|\vec{Z}\mathcal{X}\vec{x} \vdash \mathcal{X}\vec{x})$-reducible π, $\delta_\mu(\beta, \vec{x}.\alpha.\pi)$ is $(|\mu(B\vec{Z})\vec{t}| \vdash \mathcal{X}\vec{t})$-reducible.

(4) For any $(\vec{x}, \mathcal{X}\vec{x} \vdash |B|\vec{Z}\mathcal{X}\vec{x})$-reducible π, $\nu(\beta, \vec{x}.\alpha.\pi)$ is $(\mathcal{X}\vec{t} \vdash |\nu(B\vec{Z})\vec{t}|)$-reducible.

We proceed by induction on the number of logical connectives in B. The purpose of the generalization is to keep formulas $\vec{A}$ out of the picture: those are potentially large but are treated atomically in the definition of functoriality, moreover they will be interpreted by candidates $\vec{Z}$ which may not be interpretations of formulas. We first prove (3) and (4) by relying on smaller instances of (1), then we show (1) and (2) by relying on smaller instances of all four properties but also instances of (3) and (4) for an operator of the same size.

(1) We proceed by case analysis on B. When $B = \lambda\vec{p}\lambda q.q\vec{t}$, we have to establish that $F^+_{B\vec{A}}(\vec{x}.\alpha.\pi) = \lambda\beta.\pi[\vec{t}/\vec{x}][\beta/\alpha] \in |P\vec{t} \supset P'\vec{t}|$. It simply follows from Lemma 5 and the hypothesis on π. When $B = \lambda\vec{p}\lambda q.B'\vec{p}$ where q does not occur in B', we have to show $F^+_{B\vec{A}}(\vec{x}.\alpha.\pi) = \lambda\beta.\beta \in |B'\vec{Z} \supset B'\vec{Z}|$, which is trivial.
In all other cases, we use the adequacy properties and conclude by induction hypothesis. Most cases are straightforward, relying on the adequacy properties. In the implication case, i.e., B is $B_1 \supset B_2$, we use induction hypothesis (2) on B_1 and (1) on B_2. Let us only detail the least fixed point case:

$$F^+_{\lambda q.\mu(B\vec{A}q)\vec{t}}(\vec{x}.\alpha.\pi) \stackrel{def}{=} \lambda\beta.\ \delta_\mu(\beta, \vec{x}.\gamma.\mu(B\vec{A}P', \vec{x}, F^+_{\lambda q.B\vec{A}q(\mu(B\vec{A}P'))\vec{x}}(\vec{x}.\alpha.\pi)\gamma))$$

By induction hypothesis (1) with $B := \lambda\vec{p}\lambda p_{n+1}\lambda q.B\vec{p}qp_{n+1}\vec{x}$, $A_{n+1} := \mu(B\vec{A}P')$ and

[2] In (1) and (2), B has type $o^{n+1} \to o$. In (3) and (4) we are considering B of type $o^n \to (\vec{\gamma} \to o) \to (\vec{\gamma} \to o)$.

$$\dfrac{\dfrac{\dfrac{}{\ldots \vdash \beta : \nu\,(B\;P)\,\vec{t}} \qquad \dfrac{\ldots \vdash \delta_\nu(B\;P,\vec{x},\gamma) : B\;P\;(\nu\,(B\;P))\;\vec{x} \quad \ldots \vdash F^+_{(\lambda p.B\;p\;(\nu\,(B\;P))\;\vec{x})}(\vec{x}.\alpha.\pi) : B\;P\;(\nu\,(B\;P))\vec{x} \supset B\;P'\;(\nu\,(B\;P))\;\vec{x}}{\ldots,\gamma : \nu\,(B\;P)\,\vec{x} \vdash (F^*_{(\lambda p.B\;p\;(\nu\,(B\;P))\;\vec{x})}(\vec{x}.\alpha.\pi))\;\delta_\nu(B\;P,\vec{x},\gamma) : B\;P'\;(\nu\,(B\;P))\vec{x}}}{\Gamma,\beta : \nu\,(B\;P)\,\vec{t} \vdash \nu(\beta,\ldots) : \nu\,(B\;P')\;\vec{t}}}{\Gamma \vdash F^+_{\lambda p.\nu\,(B\;p)\,\vec{t}}(\vec{x}.\alpha.\pi) : \nu\,(B\;P)\,\vec{t} \supset \nu\,(B\;P')\,\vec{t}}$$

Fig. 7: Typing functoriality for greatest fixed-points

$Z_{n+1} := |\mu(B\vec{Z}\mathcal{X}')|$, we have:

$$F^+_{\ldots}(\vec{x}.\alpha.\pi) \in |B\vec{Z}\mathcal{X}(\mu(B\vec{Z}\mathcal{X}'))\vec{x} \supset B\vec{Z}\mathcal{X}'(\mu(B\vec{Z}\mathcal{X}'))\vec{x}|$$

We can now apply the $\supset$-elimination and μ-introduction principles to obtain that $\mu(B\vec{A}P',\vec{x},(F^+_{\ldots}(\vec{x}.\alpha.\pi))\gamma)$ is $(\gamma : |B\vec{Z}\mathcal{X}(\mu(B\vec{Z}\mathcal{X}'))\vec{x}| \vdash |\mu(B\vec{Z}\mathcal{X}')\vec{x}|)$-reducible. Finally, we conclude using induction hypothesis (3) with $B := \lambda\vec{p}\lambda p_{n+1}\lambda q\lambda\vec{x}.\ B\vec{p}p_{n+1}q\vec{x}$, $A_{n+1} := P$, $Z_{n+1} := \mathcal{X}$ and $\mathcal{X} := |\mu(B\vec{Z}\mathcal{X}')|$: $F^+_{\lambda q.\mu(B\vec{A}q)\vec{t}}$ is $(|\mu(B\vec{Z}\mathcal{X})\vec{t}| \vdash |\mu(B\vec{Z}\mathcal{X}')\vec{t}|)$-reducible.

(2) Antimonotonicity: symmetric of monotonicity, without the variable case.

(3) Induction: we seek to establish that $\delta_\mu(\rho,\vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t}$ when $\rho \in |\mu(B\vec{Z})\vec{t}|$ and π is $(\vec{x},|B|\vec{Z}\mathcal{X}\vec{x} \vdash \mathcal{X}\vec{x})$-reducible. We shall show that $|\mu(B\vec{Z})\vec{t}|$ is included in the set of proofs for which this holds, by showing that (a) this set is a candidate and (b) it is a prefixed point of ϕ such that $|\mu(B\vec{Z})\vec{t}| = lfp(\phi)$. Let us consider

$$\mathcal{Y} := \vec{t} \mapsto \{\ \rho \mid \delta_\mu(\rho,\vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t}\ \}$$

First, $\mathcal{Y}\vec{t}$ is a candidate for any $\vec{t}$: conditions (1) and (2) are inherited from $\mathcal{X}\vec{t}$, only condition (3) is non-trivial. Assuming that every one-step reduct of a neutral derivation ρ belongs to $\mathcal{Y}$, we prove $\delta_\mu(\rho,\vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t}$. This is done by induction on the strong normalizability of π. Using condition (3) on $\mathcal{X}\vec{t}$, it suffices to consider one-step reducts: if the reduction takes place in ρ we conclude by hypothesis; if it takes place in π we conclude by induction hypothesis; finally, it cannot take place at toplevel because ρ is neutral.
We now establish that $\phi(\mathcal{Y}) \subseteq \mathcal{Y}$: assuming $\rho \in \phi(\mathcal{Y})\vec{t}$, we show that $\delta_\mu(\rho,\vec{x}.\alpha.\pi) \in \mathcal{X}\vec{t}$. This is done by induction on the strong normalizability of ρ and π, and it suffices to show that each one step reduct belongs to $\mathcal{X}\vec{t}$, with internal reductions handled simply by induction hypothesis. Therefore we consider the case where $\rho = \mu(B\vec{A},\vec{t},\pi')$ and our derivation reduces to $\pi[\vec{t}/\vec{x}][F_{\lambda q.B\vec{A}q\vec{t}}(\vec{x}.\beta.\delta_\mu(\beta,\vec{x}.\alpha.\pi))\pi'/\alpha]$. Now, recall that $\pi[\vec{t}/\vec{x}]$ is $(|B\vec{Z}|\mathcal{X}\vec{t} \vdash \mathcal{X}\vec{t})$-reducible. Since $\mu(B\vec{A},\vec{t},\pi') = \rho \in \phi(\mathcal{Y})\vec{t}$, we also have $\pi' \in |B\vec{Z}\mathcal{Y}\vec{t}|$. By induction hypothesis (1) we obtain that $F_{\lambda q.B\vec{A}q\vec{t}}(\vec{x}.\beta.\delta_\mu(\beta,\vec{x}.\alpha.\pi))$ is $(|B\vec{Z}\mathcal{Y}\vec{t}| \vdash |B\vec{Z}\mathcal{X}\vec{t}|)$-reducible, since $\delta_\mu(\beta,\vec{x}.\alpha.\pi)$ is $(\vec{x},\beta : \mathcal{Y}\vec{x} \vdash \mathcal{X}\vec{x})$-reducible by definition of $\mathcal{Y}$. We conclude by composing all that.

(4) Coinduction is similar to induction. Let us consider

$$\begin{aligned}\mathcal{Y} := \vec{t} \mapsto \{\ &\pi \in \mathcal{SN} \mid \pi \to^* \nu(\rho,\vec{x}.\alpha.\pi) \text{ implies}\\ &\rho \in \mathcal{X}\vec{t} \text{ and } \pi \text{ is } (\vec{x},\alpha : \mathcal{X}\vec{x} \vdash |B|\vec{Z}\mathcal{X}\vec{x})\text{-reducible}\ \}\end{aligned}$$

It is easy to show that $\mathcal{Y}$ is a predicate candidate, and if we show that $\mathcal{Y} \subseteq |\nu(B\vec{Z})|$ we can conclude because the properties on ρ and π are preserved by reduction.
We have $|\nu(B\vec{Z})| = gfp(\phi)$, so it suffices to establish that $\mathcal{Y}$ is a post-fixed point of ϕ, or in other words that for any $\vec{t}$ and $\pi \in \mathcal{Y}\vec{t}$, $\delta_\nu(B\vec{A},\vec{t},\pi) \in |B|\vec{Z}\mathcal{Y}\vec{t}$. We do this as usual by induction on the strong normalizability of π and the only interesting case to consider is the toplevel reduction, which can occur when $\pi = \nu(\rho,\vec{x}.\alpha.\pi')$. The reduct is $F_{\lambda p.B\vec{A}p\vec{t}}(\vec{x}.\beta.\nu(\beta,\vec{x}.\alpha.\pi'))\ (\pi'[\vec{t}/\vec{x}][\rho/\alpha])$. It does belong to $|B\vec{Z}\mathcal{Y}\vec{t}|$ because: $\rho \in \mathcal{X}\vec{t}$ by definition of $\pi \in \mathcal{Y}\vec{t}$; $\pi'[\vec{t}/\vec{x}]$ is $(\alpha : \mathcal{X}\vec{t} \vdash |B|\vec{Z}\mathcal{X}\vec{t})$-reducible for the same reason; and finally $F_{\lambda p.\ B\vec{A}p\vec{t}}(\vec{x}.\beta.\nu(\beta,\vec{x}.\alpha.\pi')) \in |B\vec{Z}\mathcal{X}\vec{t} \supset B\vec{Z}\mathcal{Y}\vec{t}|$ by (1) since $\nu(\alpha,\vec{x}.\alpha.\pi')$ is $(\vec{x};\alpha : \mathcal{X}\vec{x} \vdash \mathcal{Y}\vec{x})$-reducible by definition of $\mathcal{Y}$.

C. *Proof of Theorem 2*

We proceed by induction on the height of π. If π is a variable, then $\pi\sigma = \sigma(\alpha)$. Thus, it belongs to $|\Gamma(\alpha)|$ by hypothesis, and since we are considering a pre-model of the congruence, and $P \equiv \Gamma(\alpha)$, we have $\pi\sigma \in |P|$.

Other cases follow from the adequacy properties established previously. For instance, if π is of the form $\lambda\alpha.\pi'$, then $P \equiv P_1 \supset P_2$ and $|P| = |P_1 \supset P_2|$. By induction hypothesis, π' is $(\Gamma,\alpha : |P_1| \vdash |P_2|)$-reducible. Equivalently, $\pi'\sigma$ is $(|P_1| \vdash |P_2|)$-reducible, and we conclude using Lemma 5. In the case where $\pi = \lambda x.\pi'$, we need to establish that each $\pi'\sigma[t/x]$ belongs to $|P'[t/x]|$. We obtain this by induction hypothesis, since $\pi'[t/x]$ has the same height as π', which is smaller than π, and we do have $\Gamma[t/x] \vdash \pi'[t/x] : P'[t/x]$. Similarly, when $\pi = \delta_=(\Gamma',\theta',\sigma',t,t',P',\pi',(\theta_i.\pi_i)_i)$, we establish $\pi\sigma \in |P'\theta'|$ by using the induction hypothesis to obtain that $\sigma'\sigma \in |\Gamma'\theta'|$, $\pi'\sigma \in |t\theta' = \vec{t}\theta'|$ and, for any i and θ'', $\pi_i\theta''$ is $(|\Gamma'\theta_i\theta''| \vdash |P'\theta_i\theta''|)$-reducible.

D. *Proof of Theorem 3*

We define $\equiv_{a_i\vec{t}}$ (resp. $\equiv_{\prec a_i\vec{t}}$) to be the congruence resulting from the extension of $\equiv$ with rule instances $a_j\vec{t'} \rightsquigarrow B$ for $a_j\vec{t'} \preceq a_i\vec{t}$ (resp. $a_j\vec{t'} \prec a_i\vec{t}$). Let us also write $P \preceq a_i\vec{t}$ (resp. $P \prec a_i\vec{t}$) when $a_j\vec{t'} \preceq a_i\vec{t}$ (resp. $a_j\vec{t'} \prec a_i\vec{t}$) for any $a_j\vec{t'}$ which may occur in P. We shall build a family of pre-models $\mathcal{M}^{a_i\vec{t}}$ such that:

(a) for any $a_i\vec{t} \prec a_j\vec{t'}$, $|a_j\vec{t'}|_{\mathcal{M}^{a_i\vec{t}}} = \mathcal{SN}$;

(b) for any $P \preceq a_j\vec{t'}$ and $a_j\vec{t'} \prec a_i\vec{t}$, $|P|_{\mathcal{M}^{a_j\vec{t'}}} = |P|_{\mathcal{M}^{a_i\vec{t}}}$;

(c) $\mathcal{M}^{a_i\vec{t}}$ is a pre-model of $\equiv_{a_i\vec{t}}$.

We proceed by well-founded induction. Assuming that $\mathcal{M}^{a_j\vec{t'}}$ is defined for all $a_j\vec{t'} \prec a_i\vec{t}$, we shall thus build $\mathcal{M}^{a_i\vec{t}}$.

We first define $\mathcal{M}^{\prec a_i\vec{t}}$ by taking each $\hat{a}_j\vec{t'}$ to be the same as in $\mathcal{M}^{a_j\vec{t'}}$ when $a_j\vec{t'} \prec a_i\vec{t}$ and $\mathcal{SN}$ otherwise. By this definition and property (b) of our pre-models, we have

$$|P|_{\mathcal{M}^{\prec a_i\vec{t}}} = |P|_{\mathcal{M}^{a_j\vec{t'}}} \text{ for any } P \preceq a_j\vec{t'} \text{ and } a_j\vec{t'} \prec a_i\vec{t}.$$

Next, we observe that $\mathcal{M}^{\prec a_i\vec{t}}$ is a pre-model of $\equiv_{\prec a_i\vec{t}}$. It suffices to check it separately for each rewrite rule. An instance $P \rightsquigarrow Q$ of a rule defining the initial congruence cannot involve the new predicates, so in that case we do have

$$|P|_{\mathcal{M}^{\prec a_i\vec{t}}} = |P|_{\mathcal{M}} = |Q|_{\mathcal{M}} = |Q|_{\mathcal{M}^{\prec a_i\vec{t}}}.$$

For a rule instance $a_j\vec{t'} \rightsquigarrow B$ with $a_j\vec{t'} \prec a_i\vec{t}$, the property is similarly inherited from $\mathcal{M}_{a_j\vec{t'}}$ because $B \preceq a_j\vec{t'}$ by (2):

$$|a_j\vec{t'}|_{\mathcal{M}^{\prec a_i\vec{t}}} = |a_j\vec{t'}|_{\mathcal{M}^{a_j\vec{t'}}} = |B|_{\mathcal{M}^{a_j\vec{t'}}} = |B|_{\mathcal{M}^{\prec a_i\vec{t}}}$$

We finally build $\mathcal{M}^{a_i\vec{t}}$ to be the same as $\mathcal{M}^{\prec a_i\vec{t}}$ except for $\hat{a}_i\vec{t}$ which is defined as follows:

- If there is no rule $a_i\vec{t''} \rightsquigarrow B$ such that $\vec{t''}\theta \equiv \vec{t}$, we define $\hat{a}_i\vec{t}$ to be $\mathcal{SN}$.
- Otherwise, pick any such B, and define $\hat{a}_i\vec{t}$ to be $|B\theta|_{\mathcal{M}^{\prec a_i\vec{t}}}$. This is uniquely defined: for any other $a_i\vec{t'} \rightsquigarrow B'$ such that $a_i\vec{t} = (a_i\vec{t'})\theta'$, we have $B\theta \equiv B'\theta'$ by (1), and thus $|B'\theta'|_{\mathcal{M}^{\prec a_i\vec{t}}} = |B\theta|_{\mathcal{M}^{\prec a_i\vec{t}}}$ since $\mathcal{M}^{\prec a_i\vec{t}}$ is *a fortiori* a pre-model of $\equiv$.

This extended pre-model satisfies (a) by construction. It is also simple to show that it satifies (b). To check that it verifies (c) we check separately each instance of a rewrite rule: by construction, our pre-model is compatible with instances of the form $a_i\vec{t} \rightsquigarrow B$, and it inherits that property from $M^{\prec a_i\vec{t}}$ for other instances.

Finally, we define our new pre-model $\mathcal{M}'$ by taking each $\hat{a}_i\vec{t}$ in $\mathcal{M}^{a_i\vec{t}}$. It is a pre-model of the extended congruence: it is easy to check that it is compatible with all rewrite rules.