An interpretation of system F through bar recursion

Valentin Blot
Queen Mary University of London

***Abstract*—There are two possible computational interpretations of second-order arithmetic: Girard's system F or Spector's bar recursion and its variants. While the logic is the same, the programs obtained from these two interpretations have a fundamentally different computational behavior and their relationship is not well understood. We make a step towards a comparison by defining the first translation of system F into a simply-typed total language with a variant of bar recursion. This translation relies on a realizability interpretation of second-order arithmetic. Due to Gödel's incompleteness theorem there is no proof of termination of system F within second-order arithmetic. However, for each individual term of system F there is a proof in second-order arithmetic that it terminates, with its realizability interpretation providing a bound on the number of reduction steps to reach a normal form. Using this bound, we compute the normal form through primitive recursion. Moreover, since the normalization proof of system F proceeds by induction on typing derivations, the translation is compositional. The flexibility of our method opens the possibility of getting a more direct translation that will provide an alternative approach to the study of polymorphism, namely through bar recursion.**

I. Introduction

Second-order λ-calculus [1], [2] is a poweful type system that allows the typing of terms such as $\lambda x.x\,x$. The language obtained is still strongly normalizing, but so far all proofs of this factrely on the notion of reducibility candidates (RCs): sets of λ-terms satisfying some axioms. In these proofs, every type has an associated RC such that every term belongs to the RC associated to its type. Then the normalization property follows as a consequence of the axioms of RCs. An important aspect of these proofs is their impredicativity: the RC associated to a universally quantified type is obtained as an intersection over all RCs. The translation presented here avoids direct reliance on the notion of RC by reducing termination of system F to termination of a variant of bar recursion, which uses an instance of Zorn's lemma.

In 1962, Spector used bar recursion [3] to extend Gödel's Dialectica interpretation of arithmetic into an interpretation of analysis, showing that bar recursion interprets the axiom scheme of comprehension. Variants of bar recursion have then been used to interpret the axioms of countable and dependent choice in a classical setting through Kreisel's modified realizability. Among these variants, modified bar recursion [4] relies on the continuity of one of its arguments to ensure termination, rather than on the explicit termination condition of original bar recursion. Krivine recently used this variant in untyped realizability for set theory [5]. The variant that we use is the BBC functional [6], that builds the family of realizers in an order that depends on the order of computation rather than on the usual order on natural numbers. While in [6], the proof of correctness of the BBC functional relies on syntactic arguments, we use an adaptation of the semantic poof of [7] that relies on Zorn's lemma. Using this we are then able to interpret the axiom scheme of comprehension, which is the only ingredient required on top of first-order arithmetic in order to get a computational interpretation of second-order arithmetic and therefore of normalization of system F.

For any single term of system F there exists a proof in second-order arithmetic that it terminates. This mapping from terms of system F to proofs of second-order arithmetic is closely related to Reynolds' abstraction theorem [8] which, as explained in [9], relies on an embedding of system F into second-order arithmetic. We use our interpretation of second-order arithmetic to extract the normal form of the system F term from its termination proof. Our technique is similar to Berger's work in the simply-typed case [10]. It is closely related to normalization by evaluation, extended to system F in [11], [12]. To avoid an encoding of λ-terms as natural numbers, we define a multi-sorted first-order logic with a sort for λ-terms with de Bruijn indices. To formalize the notion of reducibility candidates, our logic also has a sort for sets of λ-terms. Since these are first-order elements of the logic, the instantiation of a set variable with an arbitrary formula is not directly possible as in second-order arithmetic. We will however get back this possibility through our interpretation of the axiom scheme of comprehension with the BBC functional.

In a second step we fix the target programming language of the translation. This language, that we call system ΛT_{bbc}, is purely functional with a type for λ-terms, primitive recursion, and the BBC functional. System ΛT_{bbc} is in particular simply-typed and total. We also describe the sound and computationally adequate semantics of this language in the category of complete partial orders.

The last step is the definition of a realizability semantics for our logic. To each formula we associate a type of system ΛT_{bbc} and a set of realizers in the domain interpreting that type. Realizers are elements of the model rather than syntactic programs because the correctness of the BBC functional requires the existence of non-computable functions on discrete types which only exist in the model. Since we encode existential quantifications using universal ones and negation, we are able to interpret classical logic. On the other hand, the BBC functional interprets a variant of the axiom of countable choice, and its combination with our interpretation of classical logic provides a realizer of the axiom scheme of comprehension. Using this, we are then able to interpret the instantiation of set variables with arbitrary formulas. Finally, we associate for each term of system F a program that interprets the proof of its termination for weak head reduction and computes the normal form of the initial term of system F.

978-1-5090-3018-7/17/$31.00

II. Normalization of system F

In this first section, we give the details of the proof of normalization of system F that we will interpret through realizability in section V. The interested reader will find more information on this subject in e.g. [13]. Our proof uses a notion of reducibility candidates that is a variant of Tait's saturated sets [14]. We do not use Girard's reducibility candidates [15] because the corresponding normalization proof requires induction on the length of reduction of subterms when proving that the arrow construction preserves the properties of reducibility candidates. Therefore the interpretation would be much more complicated. Moreover, our version is a simplified version of Tait's saturated sets that also appears in [13] and is sufficient for weak head reduction. We make this choice again because it simplifies the interpretation, but interpreting another variant of the normalization proof is also possible.

A. Terms and substitutions

We first describe the syntax that we will be using throughout the paper. Since we are interested in interpreting formal proofs about λ-terms, we choose to represent them using de Bruijn indices, avoiding the usual complications due to α-conversion. The syntax for the set Λ of all λ-terms is as follows:

$$M, N \quad ::= \quad \underline{m} \mid \lambda.M \mid M\,N$$

where m is a natural number. We suppose that the reader is familiar with de Bruijn indices and do not recall here the translations between usual λ-terms and λ-terms with de Bruijn indices. We only give an example: the λ-term $\lambda xyz.y\,(\lambda u.x)$ is written with de Bruijn indices as $\lambda.\lambda.\lambda.\underline{1}\,(\lambda.\underline{3})$. Since we use Tait's style of reducibility candidate, we will have to manipulate λ-terms applied to an arbitrary number of arguments. We therefore also consider lists of λ-terms, for which we use the notation $\Pi = \langle M_0, \dots, M_{n-1}\rangle$. We write $M\,\Pi$ for $M\,M_0\,\dots\,M_{n-1}$ in Λ. Parallel substitution with de Bruijn indices requires the definition of a shift operation $\uparrow^k\_$ on terms. $\uparrow^k M$ is the result of incrementing the value of all variables of M with an outer index $\geq k$, and is defined as follows:

$$\uparrow^k \underline{m} = \begin{cases} \underline{m+1} & \text{if } m \geq k \\ \underline{m} & \text{otherwise} \end{cases} \qquad \begin{aligned} \uparrow^k(\lambda.M) &= \lambda.\left(\uparrow^{k+1} M\right) \\ \uparrow^k(M\,N) &= \left(\uparrow^k M\right)\left(\uparrow^k N\right) \end{aligned}$$

This operation is extended to lists of terms:

$$\uparrow^k \langle M_0, \dots, M_{n-1}\rangle = \left\langle \uparrow^k M_0, \dots, \uparrow^k M_{n-1}\right\rangle$$

We write $\uparrow M$ (resp. $\uparrow\Pi$) for $\uparrow^0 M$ (resp. $\uparrow^0\Pi$). Using the shift operation, we define parallel substitution $N\,[k \mapsto \Pi]$ where $\Pi = \langle M_0, \dots, M_{n-1}\rangle$. The result of the parallel substitution $N\,[k \mapsto \Pi]$ is obtained by substituting M_i for variables of outer index i such that $k \leq i < k+n$ in N, and subtracting n to variables of outer index $i \geq k+n$:

$$\underline{m}\,[k \mapsto \langle M_0, \dots, M_{n-1}\rangle] = \begin{cases} \underline{m} & \text{if } m < k \\ M_{m-k} & \text{if } k \leq m < k+n \\ \underline{m-n} & \text{otherwise} \end{cases}$$

$$(\lambda.M)\,[k \mapsto \Pi] = \lambda.\,(M\,[k+1 \mapsto \uparrow\Pi])$$

$$\frac{}{T_{n-1}, \dots, T_0 \vdash \underline{m} : T_m}\ {\scriptstyle 0 \leq m < n}$$

$$\frac{\Gamma, T \vdash M : U}{\Gamma \vdash \lambda.M : T \to U} \qquad \frac{\Gamma \vdash M : T \to U \qquad \Gamma \vdash N : T}{\Gamma \vdash M\,N : U}$$

$$\frac{\Gamma \vdash M : T}{\Gamma \vdash M : \forall X\,T}\ {\scriptstyle X \notin \mathrm{FV}(\Gamma)} \qquad \frac{\Gamma \vdash M : \forall X\,T}{\Gamma \vdash M : T\,\{U/X\}}$$

Fig. 1. Typing rules of system F

$$(M\,N)\,[k \mapsto \Pi] = (M\,[k \mapsto \Pi])\,(N\,[k \mapsto \Pi])$$

Substitution of a single term is defined as:

$$M\,[k \mapsto N] = M\,[k \mapsto \langle N\rangle]$$

and we write $M\,[\Pi]$ (resp. $M\,[N]$) for $M\,[0 \mapsto \Pi]$ (resp. $M\,[0 \mapsto N]$). The usual β-reduction of λ-calculus is therefore:

$$(\lambda.M)\,N \succ M\,[N]$$

The following lemma will be used in the proof of normalization:

Lemma 1.

$$M\,[k \mapsto \langle N, \Pi\rangle] = M\left[k+1 \mapsto \uparrow^k \Pi\right][k \mapsto N]$$

where $\langle N, \Pi\rangle$ is the result of prepending N to Π.

Proof. By induction on M, using $\uparrow\left(\uparrow^k\Pi\right) = \uparrow^{k+1}(\uparrow\Pi)$ for the case of a λ-abstraction. □

B. The normalization theorem

In this section we give the normalization proof of system F that we will interpret through realizability in section V. This proof is a simplified version of the usual one and only proves weak head reduction. We choose this version because it avoids some nested inductions. However, it is possible with our technique to interpret another proof since our realizability model interprets full second-order arithmetic.

First, we recall the typing rules of system F in figure 1, where types are defined as follows:

$$T, U \quad ::= \quad X \mid T \to U \mid \forall X\,T$$

where X ranges over a countable set of type variables. Since we work with de Bruijn indices, contexts are ordered lists of types (and the order is important). We use a Curry presentation (without type abstractions and applications within the terms) since it simplifies the syntax and we are not interested into type checking or inference. As explained above, we only consider weak head reduction:

$$(\lambda.M)\,N\,\Pi \succ M\,[N]\,\Pi$$

and write $M{\downarrow}$ if M normalizes for the above reduction.

The normalization proof goes as follows: first, we define the set $\mathcal{RC} \subseteq \mathcal{P}(\Lambda)$ of reducibility candidates and we prove that the set of normalizing terms is a reducibility candidate. Then, we associate a set $RC_{T,v} \subseteq \Lambda$ to each type T of system F with valuation $v : \mathrm{FV}(T) \to \mathcal{RC}$, and we prove that $RC_{T,v}$ is a reducibility candidate. Finally, we prove that if a closed term

M is of closed type T, then $M \in RC_{T,\emptyset}$. Since one of the properties of reducibility candidates is that they contain only normalizing terms, we can then conclude that M normalizes.

We now give the proof in more details. First, define the set $\mathcal{RC}$ of reducibility candidates:

Definition 1 (Reducibility candidate)**.** $\mathfrak{X} \subseteq \Lambda$ *is in* $\mathcal{RC}$ *if:*

- *For any list of terms* Π*, we have* $\underline{0}\,\Pi \in \mathfrak{X}$
- *If* $M \in \mathfrak{X}$*, then* $M{\downarrow}$
- *If* $M\,[N]\,\Pi \in \mathfrak{X}$*, then* $(\lambda.M)\,N\,\Pi \in \mathfrak{X}$

In particular, the set of normalizing terms is a reducibility candidate:

Lemma 2. $\{M \in \Lambda \mid M{\downarrow}\} \in \mathcal{RC}$

Proof.
- For any Π, $\underline{0}\,\Pi$ is in head normal form so $\underline{0}\,\Pi{\downarrow}$
- If $M{\downarrow}$, then $M{\downarrow}$
- If $M\,[N]\,\Pi{\downarrow}$, then $(\lambda.M)\,N\,\Pi{\downarrow}$ because:

$$(\lambda.M)\,N\,\Pi \succ M\,[N]\,\Pi \qquad \square$$

As explained above, in the second step we define a set $RC_{T,v} \subseteq \Lambda$ for each type T with valuation v:

Definition 2. *If* T *is a type of system F and if* $v : FV(T) \to \mathcal{RC}$*, we define* $RC_{T,v}$ *inductively:*

- $RC_{X,v} = v(X)$
- $RC_{T\to U,v} = \{M \mid \forall N \in RC_{T,v}, M\,N \in RC_{U,v}\}$
- $RC_{\forall X\,T,v} = \bigcap \left\{RC_{T,v\uplus\{X\mapsto\mathfrak{X}\}} \;\middle|\; \mathfrak{X} \in \mathcal{RC}\right\}$

These sets are indeed reducibility candidates:

Lemma 3. *If* T *is a type and* $v : FV(T) \to \mathcal{RC}$*, then* $RC_{T,v} \in \mathcal{RC}$

Proof. By induction on T:

- Since $v(X) \in \mathcal{RC}$, we have $RC_{X,v} = v(X) \in \mathcal{RC}$
- Suppose $RC_{T,v} \in \mathcal{RC}$ and $RC_{U,v} \in \mathcal{RC}$.
  - If Π is a list of terms and $M \in RC_{T,v}$ then $\langle \Pi, M\rangle$ (result of appending M to Π) is a list of terms so:
  $$(\underline{0}\,\Pi)\,M = \underline{0}\,\langle \Pi, M\rangle \in RC_{U,v}$$
  by induction hypothesis
  - Let $M \in RC_{T\to U,v}$. We have $\underline{0} = \underline{0}\,\langle\rangle \in RC_{T,v}$ by induction hypothesis, so $M\,\underline{0} \in RC_{U,v}$ by definition of $RC_{T\to U,v}$ and $M\,\underline{0}{\downarrow}$ by induction hypothesis. Since any reduction step of M can be turned into a reduction step of $M\,\underline{0}$, we get $M{\downarrow}$
  - Suppose $M\,[N]\,\Pi \in RC_{T\to U,v}$. Then for any $P \in RC_{T,v}$ we have by definition of $RC_{T\to U,v}$:
  $$M\,[N]\,\langle \Pi, P\rangle = (M\,[N]\,\Pi)\,P \in RC_{U,v}$$
  and therefore:
  $$(\lambda.M)\,N\,\Pi\,P = (\lambda.M)\,N\,\langle \Pi, P\rangle \in RC_{U,v}$$
  by induction hypothesis
- Suppose $RC_{T,v\uplus\{X\mapsto\mathfrak{X}\}} \in \mathcal{RC}$ for every $\mathfrak{X} \in \mathcal{RC}$.
  - If Π is a list of terms, then $\underline{0}\,\Pi \in RC_{T,v\uplus\{X\mapsto\mathfrak{X}\}}$ for every $\mathfrak{X} \in \mathcal{RC}$ by induction hypothesis
  - If $M \in RC_{\forall X\,T,v}$, then $M \in RC_{T,v\uplus\{X\mapsto\{N \mid N{\downarrow}\}\}}$ since $\{N \mid N{\downarrow}\} \in \mathcal{RC}$ by lemma 2, and therefore $M{\downarrow}$ by induction hypothesis
  - If $M\,[N]\,\Pi \in RC_{\forall X\,T,v}$ and $\mathfrak{X} \in \mathcal{RC}$, then:
  $$M\,[N]\,\Pi \in RC_{T,v\uplus\{X\mapsto\mathfrak{X}\}}$$
  and therefore:
  $$(\lambda.M)\,N\,\Pi \in RC_{T,v\uplus\{X\mapsto\mathfrak{X}\}}$$
  by induction hypothesis $\square$

In the last step of the normalization proof, we prove that each term of system F belongs to the reducibility candidate associated to its type:

Lemma 4. *If* $T_{n-1}, \ldots, T_0 \vdash N : U$ *in system F and if* $v : FV(T_{n-1}, \ldots T_0, U) \to \mathcal{RC}$ *and* $\Pi = \langle M_0, \ldots, M_{n-1}\rangle$ *are such that* $M_i \in RC_{T_i,v}$ *for* $0 \le i < n$*, then* $N\,[\Pi] \in RC_{U,v}$

Proof. By induction on the typing derivation:

- $T_{n-1}, \ldots, T_0 \vdash \underline{m} : T_m$: we have $\underline{m}\,[\Pi] = M_m \in RC_{T_m,v}$ as an hypothesis
- $T_{n-1}, \ldots, T_0 \vdash \lambda.N : U \to V$: if $P \in RC_{U,v}$, we have:
$$N\,[1 \mapsto \uparrow\Pi]\,[P] = N\,[\langle P, \Pi\rangle] \in RC_{V,v}$$
by lemma 1 and induction hypothesis, so:
$$(\lambda.N)\,[\Pi]\,P = \lambda.\,(N\,[1 \mapsto \uparrow\Pi])\,P \in RC_{V,v}$$
by definition of reducibility candidates
- $T_{n-1}, \ldots, T_0 \vdash N\,P : V$: we have:
$$(N\,P)\,[\Pi] = N\,[\Pi]\,P\,[\Pi] \in RC_{V,v}$$
because $N\,[\Pi] \in RC_{U\to V,v}$ and $P\,[\Pi] \in RC_{U,v}$ by induction hypothesis
- $T_{n-1}, \ldots, T_0 \vdash N : \forall X\,U$: if $\mathfrak{X} \in \mathcal{RC}$ then we have $M_i \in RC_{T_i,v\uplus\{X\mapsto\mathfrak{X}\}}$ because $X \notin \mathrm{FV}(T_i)$, and therefore $N\,[\Pi] \in RC_{U,v\uplus\{X\mapsto\mathfrak{X}\}}$ by induction hypothesis
- $T_{n-1}, \ldots, T_0 \vdash N : U\,\{V/X\}$: we have $RC_{V,v} \in \mathcal{RC}$ by lemma 3, so:
$$N\,[\Pi] \in RC_{U,v\uplus\{X\mapsto RC_{V,v}\}} = RC_{U\{V/X\},v}$$
by induction hypothesis (the equality between these two reducibility candidates is proved by induction on U) $\square$

We can now conclude our normalization proof of system F:

Theorem 1. *If a closed term* M *has closed type* T *in system F, then* $M{\downarrow}$

Proof. Lemma 4 gives $M \in RC_{T,\emptyset}$ and we get $M{\downarrow}$ by lemma 3 and by definition of reducibility candidates. $\square$

III. A LOGIC FOR λ-TERMS

This section is devoted to the definition of a first-order multi-sorted logic in which we can easily formalize the normalization proof of system F described in the previous section. The main property is that our logic directly manipulates λ-terms rather than representing them through an encoding into natural numbers.

A. Definitions

Since our realizability interpretation will be simply-typed, we use a first-order representation of second-order arithmetic. In particular, sets of λ-terms (and reducibility candidates) are first-order citizens and we cannot instantiate them with arbitrary formulas. We will however get back this possibility in the next section through an interpretation of the axiom scheme of comprehension with the BBC functional. The logic has sorts for natural numbers, λ-terms, lists of λ-terms, sets of λ-terms and booleans. We distinguish elements of different sorts by using diffferent notations:

$$\begin{array}{rcl} m & ::= & i \mid 0 \mid S\, m \\ M & ::= & t \mid \underline{m} \mid \lambda.M \mid M\,\Pi \mid M\,[\Pi] \\ \Pi & ::= & \pi \mid \langle\rangle \mid \langle \Pi, M\rangle \qquad X \quad ::= \quad X \\ \Phi & ::= & b \mid tt \mid ff \mid M \in X \mid M{\downarrow}^m \end{array}$$

where i, t, π, X and b range over countable sets of sorted variables of the logic. Notations m, M, Π and Φ are used as meta-variables ranging over the terms of the logic. Since the only terms of sort "set" (ranged over with X) are variables, the meta-variables of sort "set" are exactly the variables of the logic and we do not need any specific notation. The terms Φ are booleans that reflect validity. Note that in $M \in X$ (resp. $M{\downarrow}^m$), $\in$ (resp. $\downarrow$) is a binary function symbol taking a term M and a set X (resp. a term M and a natural number m) and returning a boolean value. $M{\downarrow}^m$ means that M can reduce for m steps of weak head reduction without reaching a normal form. We abbreviate $\langle\langle \ldots \langle\langle\rangle, M_0\rangle, \ldots\rangle, M_{n-1}\rangle$ as $\langle M_0, \ldots, M_{n-1}\rangle$ and $M\,[\langle N\rangle]$ as $M\,[N]$. Formulas are defined as follows:

$$A, B \quad ::= \quad \Phi \mid A \Rightarrow B \mid A \wedge B \mid \forall \epsilon\, A$$

where ϵ ranges over variables of any sort: i, t, π, X, b. The predicate Φ should be thought of as "$\Phi = tt$". We also define the following abbreviations:

$$\neg A \triangleq A \Rightarrow ff \qquad \exists \epsilon\, A \triangleq \neg\forall\epsilon\neg A \qquad M{\downarrow} \triangleq \neg\forall i\, M{\downarrow}^i$$
$$A \Leftrightarrow B \triangleq (A \Rightarrow B) \wedge (B \Rightarrow A)$$

where ϵ ranges over variables of any sort. Note that our logic does not contain primitive existential quantifications. This is because we need classical logic for interpreting the axiom scheme of comprehension, and therefore we choose to work in a subset of the logic corresponding to the target of Gödel's negative translation. This is to be contrasted with the dialectica-like interpretations that perform an explicit negative translation from classical to intuitionistic logic, before giving a computational interpretation of the target of the translation.

We also define the notion of dependent formulas that will be useful to our formalization of the normalization proof. A 1-formula is a formula depending on elements of the logic. For example, $A\,(M, \Phi) \equiv \forall\pi\,(M\,\pi \in X \Rightarrow \Phi)$ is a formula depending on a term M and a boolean Φ (containing moreover a free variable X). We avoid the capture of bound variables, so $A\,(t\,\pi, t \in X)$ is $\forall\pi'\,(t\,\pi\,\pi' \in X \Rightarrow t \in X)$. We also consider 2-formulas: formulas depending on 1-formulas. The only 2-formulas that we consider depend on one 1-formula which itself depends on one term. An example of 2-formula is $A\,(B) \equiv \forall\pi\,(B\,(t\,\pi) \Rightarrow \underline{0} \in X)$. Again, we avoid the capture of bound variables: if $B\,(M) \equiv M\,\pi \in X \Rightarrow M \in X$, then $A\,(B)$ is $\forall\pi'\,((t\,\pi'\,\pi \in X \Rightarrow t\,\pi' \in X) \Rightarrow \underline{0} \in X)$.

For each variable X of sort set we define the 1-formula $\overline{X}\,(M) \equiv M \in X$. We also define the 1-formula $\Downarrow\,(M) \equiv M{\downarrow}$. If A is a formula and X is a variable of sort set, then we write $\overline{X} \mapsto A$ for the 2-formula such that $\left(\overline{X} \mapsto A\right)(B)$ is A where every atom of the form $M \in X$ has been replaced with $B\,(M)$.

We also define the 2-formula $\mathcal{R}ed\mathcal{C}and\,(A)$ which says that the set of M such that $A\,(M)$ holds is a reducibility candidate:

$$\begin{array}{rl} \mathcal{R}ed\mathcal{C}and\,(A) \triangleq & (\forall\pi\, A\,(\underline{0}\,\pi) \wedge \forall t\,(A\,(t) \Rightarrow t{\downarrow})) \\ & \wedge \forall t\,\forall u\,\forall\pi\,(A\,(t\,[u]\,\pi) \Rightarrow A\,((\lambda.t)\,\langle u\rangle\,\pi)) \end{array}$$

Finally, to each type T of system F built from variables X of our logic we associate the 1-formula RC_T defined by induction:

$$RC_X \triangleq \overline{X} \quad RC_{T\to U}\,(M) \triangleq \forall t\,(RC_T\,(t) \Rightarrow RC_U\,(M\,\langle t\rangle))$$
$$RC_{\forall X\,T}\,(M) \triangleq \forall X\,\left(\mathcal{R}ed\mathcal{C}and\,\left(\overline{X}\right) \Rightarrow RC_T\,(M)\right)$$

It is easy to see that the free variables of sort set in $RC_T\,(M)$ are exactly the free variables of T.

B. Stating normalization

We give now the main formulas that we will have to realize in order to get a computational interpretation of the normalization theorem. First, our logic is first-order. Therefore, in order to interpret the instantiation of arbitrary formulas for set variables we must first interpret the axiom scheme of comprehension. If A is a 1-formula with one parameter of sort term and if X is not a free set variable of $A\,(M)$, then the corresponding instance of comprehension is:

$$\exists X\,\forall t\,(t \in X \Leftrightarrow A\,(t))$$

This scheme will be interpreted using the BBC functional. Then, using comprehension, we will interpret the first-order equivalent of the elimination of second-order quantification. If A is a 2-formula, B is a 1-formula with one parameter of sort term and $X \notin \mathrm{FV}\,(A)$ (meaning that $X \notin \mathrm{FV}\,(A\,(C))$ whenever $X \notin \mathrm{FV}\,(C)$), then this elimination is:

$$\forall X\, A\,\left(\overline{X}\right) \Rightarrow A\,(B)$$

Interpreting this family of implications from the axiom scheme of comprehension requires the definition of a realizer by induction on A. The interpretation of the instantiation of set variables with arbitrary formulas provides us with an interpretation of full second-order arithmetic. Building on this, we then interpret the formalization of lemma 2 in our logic:

$$\mathcal{R}ed\mathcal{C}and\,(\Downarrow)$$

The second step is the interpretation of the formalization of lemma 3. If $\mathrm{FV}\,(T) \subseteq \{X_0, \ldots, X_{n-1}\}$ then this is:

$$\begin{array}{rl} \forall X_0(\mathcal{R}ed\mathcal{C}and\,\left(\overline{X_0}\right) \Rightarrow \ldots \Rightarrow \forall X_{n-1}(\mathcal{R}ed\mathcal{C}and\,\left(\overline{X_{n-1}}\right) & \\ \Rightarrow \mathcal{R}ed\mathcal{C}and\,(RC_T))\ldots) & \end{array}$$

The last step consists of interpreting the formalization of lemma 4. If $T_0, \ldots, T_{m-1}, U$ are types such that $\mathrm{FV}(T_0, \ldots, T_{m-1}, U) \subseteq \{X_0, \ldots, X_{n-1}\}$ and if $T_{m-1}, \ldots, T_0 \vdash M : U$ is the conclusion of a valid typing derivation in system F, then this is:

$$\begin{aligned}\forall X_0(\mathcal{R}ed\mathcal{C}and\left(\overline{X_0}\right) \Rightarrow \ldots \Rightarrow \forall X_{n-1}(\mathcal{R}ed\mathcal{C}and\left(\overline{X_{n-1}}\right)\\ \Rightarrow \forall t_{m-1}(RC_{T_{m-1}}(t_{m-1}) \Rightarrow \ldots \Rightarrow \forall t_0(RC_{T_0}(t_0)\\ \Rightarrow RC_U(M[\langle t_0, \ldots, t_{m-1}\rangle])) \ldots)) \ldots)\end{aligned}$$

The interpretation of the formula above provides a realizer of $RC_T(M)$ for each closed term M of closed type T in system F, from which we will extract a bound on the number of reduction steps needed for reaching a normal form. Finally, we will use this extracted bound to compute the normal form of M using primitive recursion.

IV. A simply-typed programming language with the BBC functional

In this section, we define the target of our translation of system F: a simply-typed functional programming language that we call system ΛT_{bbc}. This language has product types, basic types for natural numbers, λ-terms and lists of λ-terms, primitive recursion on these basic types and the BBC functional. We also give a domain-theoretic denotational semantics for this programming language that is sound and computationally adequate.

A. Syntax of system ΛT_{bbc}

We first define system ΛT, and then extend it to system ΛT_{bbc} by adding the BBC functional together with its reduction rule. The programming language system ΛT is an extension of Gödel's system T with types for λ-terms and lists of λ-terms, together with primitive recursion on these. The types of system ΛT are defined by the following grammar:

$$\boldsymbol{\sigma}, \boldsymbol{\tau} \quad ::= \quad \boldsymbol{\iota} \mid \boldsymbol{\lambda} \mid \boldsymbol{\lambda}^* \mid \boldsymbol{\sigma} \to \boldsymbol{\tau} \mid \boldsymbol{\sigma} \times \boldsymbol{\tau}$$

where $\boldsymbol{\iota}$ is the type of natural numbers, $\boldsymbol{\lambda}$ is the type of λ-terms, $\boldsymbol{\lambda}^*$ is the type of lists of λ-terms, $\boldsymbol{\sigma} \to \boldsymbol{\tau}$ is the type of functions from $\boldsymbol{\sigma}$ to $\boldsymbol{\tau}$ and $\boldsymbol{\sigma} \times \boldsymbol{\tau}$ is the product type of $\boldsymbol{\sigma}$ to $\boldsymbol{\tau}$. The syntax of system ΛT is given along with its typing rules in figure 2 and its reduction rules are given in figure 3. Note that we take the convention of writing lists with the most recent element at the end since the addition of an element to a list corresponds to the extension of an applicative context with one more argument. We use iterators rather than recursors only for simplicity: recursors can nevertheless be defined using iterators and product types. Using iterators we can define a generalized application $\mathtt{app}^* : \boldsymbol{\lambda} \to \boldsymbol{\lambda}^* \to \boldsymbol{\lambda}$ such that if $\mathtt{M}, \mathtt{N}_0, \ldots, \mathtt{N}_{n-1} \in \boldsymbol{\lambda}$:

$$\begin{aligned}\mathtt{app}^*\,\mathtt{M}(\mathtt{cons}\,(\mathtt{cons}\,(\ldots\mathtt{cons}\,\mathtt{nil}\,\mathtt{N}_0\ldots)\,\mathtt{N}_{n-1}))\\ \rightsquigarrow^* \mathtt{app}\,(\ldots(\mathtt{app}\,\mathtt{M}\,\mathtt{U}_0)\ldots)\,\mathtt{U}_{n-1}\end{aligned}$$

where $\mathtt{N}_i \rightsquigarrow^* \mathtt{U}_i$. We can also define a shift operation on lists of terms $\mathtt{shift}^* : \boldsymbol{\lambda}^* \to \boldsymbol{\lambda}^*$ implementing the operation $\uparrow$ described in section II-A. Finally, we can define a substitution

$$\frac{}{\Gamma, x : \boldsymbol{\sigma} \vdash x : \boldsymbol{\sigma}} \qquad \frac{}{\Gamma \vdash \mathtt{c} : \boldsymbol{\sigma}}\ (\mathtt{c}:\boldsymbol{\sigma}) \in \mathcal{C}st$$

$$\frac{\Gamma, x : \boldsymbol{\sigma} \vdash \mathtt{M} : \boldsymbol{\tau}}{\Gamma \vdash \lambda x.\mathtt{M} : \boldsymbol{\sigma} \to \boldsymbol{\tau}} \qquad \frac{\Gamma \vdash \mathtt{M} : \boldsymbol{\sigma} \to \boldsymbol{\tau} \qquad \Gamma \vdash \mathtt{N} : \boldsymbol{\sigma}}{\Gamma \vdash \mathtt{MN} : \boldsymbol{\tau}}$$

$$\frac{\Gamma \vdash \mathtt{M} : \boldsymbol{\sigma} \qquad \Gamma \vdash \mathtt{N} : \boldsymbol{\tau}}{\Gamma \vdash \langle \mathtt{M}, \mathtt{N} \rangle : \boldsymbol{\sigma} \times \boldsymbol{\tau}}$$

$$\frac{\Gamma \vdash \mathtt{M} : \boldsymbol{\sigma} \times \boldsymbol{\tau}}{\Gamma \vdash \mathtt{p}_1\,\mathtt{M} : \boldsymbol{\sigma}} \qquad \frac{\Gamma \vdash \mathtt{M} : \boldsymbol{\sigma} \times \boldsymbol{\tau}}{\Gamma \vdash \mathtt{p}_2\,\mathtt{M} : \boldsymbol{\tau}}$$

where $\mathcal{C}st$ is:

$$\mathtt{z} : \boldsymbol{\iota} \qquad \mathtt{s} : \boldsymbol{\iota} \to \boldsymbol{\iota} \qquad \mathtt{it}_{\boldsymbol{\iota}} : \boldsymbol{\sigma} \to (\boldsymbol{\sigma} \to \boldsymbol{\sigma}) \to \boldsymbol{\iota} \to \boldsymbol{\sigma}$$

$$\mathtt{var} : \boldsymbol{\iota} \to \boldsymbol{\lambda} \qquad \mathtt{abs} : \boldsymbol{\lambda} \to \boldsymbol{\lambda} \qquad \mathtt{app} : \boldsymbol{\lambda} \to \boldsymbol{\lambda} \to \boldsymbol{\lambda}$$

$$\mathtt{it}_{\boldsymbol{\lambda}} : (\boldsymbol{\iota} \to \boldsymbol{\sigma}) \to (\boldsymbol{\sigma} \to \boldsymbol{\sigma}) \to (\boldsymbol{\sigma} \to \boldsymbol{\sigma} \to \boldsymbol{\sigma}) \to \boldsymbol{\lambda} \to \boldsymbol{\sigma}$$

$$\mathtt{nil} : \boldsymbol{\lambda}^* \qquad \mathtt{cons} : \boldsymbol{\lambda}^* \to \boldsymbol{\lambda} \to \boldsymbol{\lambda}^*$$

$$\mathtt{it}_{\boldsymbol{\lambda}^*} : \boldsymbol{\sigma} \to (\boldsymbol{\sigma} \to \boldsymbol{\lambda} \to \boldsymbol{\sigma}) \to \boldsymbol{\lambda}^* \to \boldsymbol{\sigma}$$

Fig. 2. Typing rules of system ΛT

$$\frac{}{(\lambda x.\mathtt{M})\,\mathtt{N} \rightsquigarrow \mathtt{M}\{\mathtt{N}/x\}}$$

$$\frac{}{\mathtt{p}_1\,\langle \mathtt{M}, \mathtt{N} \rangle \rightsquigarrow \mathtt{M}} \qquad \frac{}{\mathtt{p}_2\,\langle \mathtt{M}, \mathtt{N} \rangle \rightsquigarrow \mathtt{N}}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\iota}}\,\mathtt{MN}\,\mathtt{z} \rightsquigarrow \mathtt{M}} \qquad \frac{}{\mathtt{it}_{\boldsymbol{\iota}}\,\mathtt{MN}\,(\mathtt{s}\,\mathtt{U}) \rightsquigarrow \mathtt{N}\,(\mathtt{it}_{\boldsymbol{\iota}}\,\mathtt{MN}\,\mathtt{U})}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,(\mathtt{var}\,\mathtt{U}) \rightsquigarrow \mathtt{MU}}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,(\mathtt{abs}\,\mathtt{U}) \rightsquigarrow \mathtt{N}\,(\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,\mathtt{U})}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,(\mathtt{app}\,\mathtt{UV}) \rightsquigarrow \mathtt{P}\,(\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,\mathtt{U})\,(\mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,\mathtt{V})}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\lambda}^*}\,\mathtt{MN}\,\mathtt{nil} \rightsquigarrow \mathtt{M}}$$

$$\frac{}{\mathtt{it}_{\boldsymbol{\lambda}^*}\,\mathtt{MN}\,(\mathtt{cons}\,\mathtt{UV}) \rightsquigarrow \mathtt{N}\,(\mathtt{it}_{\boldsymbol{\lambda}^*}\,\mathtt{MN}\,\mathtt{U})\,\mathtt{V}}$$

$$\frac{\mathtt{M} \rightsquigarrow \mathtt{N}}{E[\mathtt{M}] \rightsquigarrow E[\mathtt{N}]}$$

where:

$$\begin{aligned}\mathtt{U}, \mathtt{V} \quad ::= \quad & \mathtt{z} \mid \mathtt{s}\,\mathtt{U} \mid \mathtt{var}\,\mathtt{U} \mid \mathtt{abs}\,\mathtt{U} \mid \mathtt{app}\,\mathtt{UV} \mid \mathtt{nil} \mid \mathtt{cons}\,\mathtt{UV}\\ E[\_] \quad ::= \quad & [\_] \mid E[\_]\,\mathtt{M} \mid \mathtt{p}_1\,E[\_] \mid \mathtt{p}_2\,E[\_]\\ & \mid \mathtt{s}\,E[\_] \mid \mathtt{it}_{\boldsymbol{\iota}}\,\mathtt{MN}\,E[\_]\\ & \mid \mathtt{var}\,E[\_] \mid \mathtt{abs}\,E[\_] \mid \mathtt{app}\,E[\_] \mid \mathtt{app}\,\mathtt{U}\,E[\_]\\ & \mid \mathtt{it}_{\boldsymbol{\lambda}}\,\mathtt{MNP}\,E[\_]\\ & \mid \mathtt{nil} \mid \mathtt{cons}\,E[\_] \mid \mathtt{cons}\,\mathtt{U}\,E[\_] \mid \mathtt{it}_{\boldsymbol{\lambda}^*}\,\mathtt{MN}\,E[\_]\end{aligned}$$

Fig. 3. Reductions in system ΛT

operation $\mathtt{subst} : \boldsymbol{\lambda} \to \boldsymbol{\iota} \to \boldsymbol{\lambda}^* \to \boldsymbol{\lambda}$ implementing the operation $\_[\_ \mapsto \_]$ described in section II-A.

We now extend ΛT with a functional that interprets the axiom scheme of comprehension on λ-terms. The lack of a canonical ordering on λ-terms is our main motivation for choosing the BBC functional rather than modified bar recursion. Before introducing the BBC functional on $\boldsymbol{\lambda}$ we need a notion of partial functions on $\boldsymbol{\lambda}$ that we can test for definedness at any value, so we now describe our encoding of these in system ΛT. We define the type of these functions as:

$$\boldsymbol{\sigma}^\dagger \triangleq \boldsymbol{\lambda} \to \boldsymbol{\iota} \times \boldsymbol{\sigma}$$

with the convention that $\mathtt{M} : \boldsymbol{\sigma}^\dagger$ is defined at $\mathtt{N} : \boldsymbol{\lambda}$ if

$\mathtt{p}_1(\mathtt{M}\,\mathtt{N}) \rightsquigarrow^* \mathtt{z}$, in which case its value is $\mathtt{p}_2(\mathtt{M}\,\mathtt{N})$, and undefined otherwise. In order to define the empty function we need to have a canonical element $\mathtt{can}_{\boldsymbol{\sigma}}$ at every type $\boldsymbol{\sigma}$, defined inductively as follows:

$$\mathtt{can}_{\boldsymbol{\iota}} \triangleq \mathtt{z} \qquad \mathtt{can}_{\boldsymbol{\lambda}} \triangleq \mathtt{var}\,\mathtt{z} \qquad \mathtt{can}_{\boldsymbol{\lambda}^*} \triangleq \mathtt{nil}$$
$$\mathtt{can}_{\boldsymbol{\sigma}\to\boldsymbol{\tau}} \triangleq \lambda\_.\mathtt{can}_{\boldsymbol{\tau}} \qquad \mathtt{can}_{\boldsymbol{\sigma}\times\boldsymbol{\tau}} \triangleq \langle \mathtt{can}_{\boldsymbol{\sigma}}, \mathtt{can}_{\boldsymbol{\tau}} \rangle$$

We write $\{\} : \boldsymbol{\sigma}^\dagger$ for the strict function with empty support:

$$\{\} \triangleq \mathtt{it}_{\boldsymbol{\lambda}}\,(\lambda\_.\langle \mathtt{s}\,\mathtt{z}, \mathtt{can}_{\boldsymbol{\sigma}}\rangle)\,(\lambda\_.\langle \mathtt{s}\,\mathtt{z}, \mathtt{can}_{\boldsymbol{\sigma}}\rangle)\,(\lambda\_\_.\langle \mathtt{s}\,\mathtt{z}, \mathtt{can}_{\boldsymbol{\sigma}}\rangle)$$

which is such that $\{\}\,\mathtt{M} \rightsquigarrow^* \langle \mathtt{s}\,\mathtt{z}, \mathtt{can}_{\boldsymbol{\sigma}}\rangle$ for any $\mathtt{M} : \boldsymbol{\lambda}$, that is, $\{\}$ is the everywhere undefined function. The strictness of $\{\}$ will be necessary in the proof of correctness of the BBC functional in section V-D. We also write $\mathtt{M} \mid \mathtt{N} : \boldsymbol{\lambda} \to \boldsymbol{\sigma}$ for the completion of $\mathtt{M} : \boldsymbol{\sigma}^\dagger$ with $\mathtt{N} : \boldsymbol{\lambda} \to \boldsymbol{\sigma}$, defined as:

$$\mathtt{M} \mid \mathtt{N} \triangleq \lambda x.\mathtt{it}_{\boldsymbol{\iota}}\,(\mathtt{p}_2\,(\mathtt{M}\,x))\,(\lambda\_.\mathtt{N}\,x)\,(\mathtt{p}_1\,(\mathtt{M}\,x))$$

and which is such that:

$$(\mathtt{M} \mid \mathtt{N})\,\mathtt{P} \rightsquigarrow^* \begin{cases} \mathtt{p}_2\,(\mathtt{M}\,\mathtt{P}) & \text{if } \mathtt{p}_1\,(\mathtt{M}\,\mathtt{P}) \rightsquigarrow^* \mathtt{z} \\ \mathtt{N}\,\mathtt{P} & \text{otherwise} \end{cases}$$

Since equality on $\boldsymbol{\lambda}$ is decidable in system ΛT (that is, there exists a term $\mathtt{M} : \boldsymbol{\lambda} \to \boldsymbol{\lambda} \to \boldsymbol{\iota}$ such that for any $\mathtt{N}, \mathtt{P} : \boldsymbol{\lambda}$, $\mathtt{M}\,\mathtt{N}\,\mathtt{P} \rightsquigarrow^* \mathtt{z}$ if and only if $\mathtt{N} \rightsquigarrow^* \mathtt{U}$ and $\mathtt{P} \rightsquigarrow^* \mathtt{U}$ for some $\mathtt{U}$), we can also define $\mathtt{M} \cup \{\mathtt{N} \mapsto \mathtt{P}\} : \boldsymbol{\sigma}^\dagger$ that overwrites/extends $\mathtt{M} : \boldsymbol{\sigma}^\dagger$ with value $\mathtt{P} : \boldsymbol{\sigma}$ at input $\mathtt{N} : \boldsymbol{\lambda}$, that is:

$$(\mathtt{M} \cup \{\mathtt{N} \mapsto \mathtt{P}\})\,\mathtt{Q} \rightsquigarrow^* \begin{cases} \langle \mathtt{z}, \mathtt{P}\rangle & \text{if } \mathtt{N} \rightsquigarrow^* \mathtt{U} \text{ and } \mathtt{Q} \rightsquigarrow^* \mathtt{U} \\ \mathtt{M}\,\mathtt{Q} & \text{otherwise} \end{cases}$$

With this new tool we can now define the BBC functional:

$$\mathtt{bbc} : ((\boldsymbol{\sigma} \to \boldsymbol{\iota}) \to \boldsymbol{\sigma}) \to ((\boldsymbol{\lambda} \to \boldsymbol{\sigma}) \to \boldsymbol{\iota}) \to \boldsymbol{\sigma}^\dagger \to \boldsymbol{\iota}$$

together with its reduction rule:

$$\mathtt{bbc}\,\mathtt{M}\,\mathtt{N}\,\mathtt{P} \rightsquigarrow \mathtt{N}\,(\mathtt{P} \mid \lambda y.\mathtt{M}\,(\lambda z.\mathtt{bbc}\,\mathtt{M}\,\mathtt{N}\,(\mathtt{P} \cup \{y \mapsto z\})))$$

System ΛT_{bbc} is obtained by extending system ΛT with the constant $\mathtt{bbc}$ together with its reduction rule.

B. Continuous semantics of system ΛT_{bbc}

Because the proof of correctness of the BBC functional requires the existence of non-computable realizers, our realizers will be elements of a model of system ΛT_{bbc} rather than mere programs. Since system ΛT_{bbc} can be seen as a subset of PCF where recursion is restricted to primitive recursion and the BBC functional, it is natural to consider a domain-theoretic semantics. More precisely, we will define a denotational semantics of system ΛT_{bbc} in complete partial orders. First, we recall some basic definitions:

Definition 3 (complete partial order)**.** *A partial order $(D, \leq)$ is a complete partial order (cpo) if:*

- *D has a least element $\bot$*
- *Every directed subset Δ of D has a least upper bound $\sqcup\Delta$, where $\Delta \subseteq D$ is directed if it is non-empty and:*

$$\forall \varphi \in \Delta\, \forall \psi \in \Delta\, \exists \theta \in \Delta\, (\varphi \leq \theta \wedge \psi \leq \theta)$$

Definition 4 (continuous function)**.** *If $(D, \leq)$ and $(E, \leq)$ are cpos, a function $\varphi : D \to E$ is continuous if for every directed subset Δ of D, $\varphi(\Delta)$ is directed and:*

$$\varphi(\sqcup\Delta) = \sqcup\varphi(\Delta)$$

Definition 5 (product of cpos)**.** *If $(D, \leq)$ and $(E, \leq)$ are cpos, then $D \times E$ is a cpo for the pointwise ordering:*

$$(\varphi, \psi) \leq (\varphi', \psi') \Leftrightarrow (\varphi \leq \varphi' \wedge \psi \leq \psi')$$

The projection functions from $D \times E$ to D and E will be written π_1 and π_2.

Definition 6 (cpo of continuous functions)**.** *If $(D, \leq)$ and $(E, \leq)$ are cpos, then the set of continuous functions from D to E is a cpo for the pointwise ordering:*

$$\varphi \leq \varphi' \Leftrightarrow \forall \psi \in D\, (\varphi(\psi) \leq \varphi'(\psi))$$

Definition 7 (flat cpo)**.** *If X is a set, then $X_\bot = X \cup \{\bot\}$ is a cpo for the following ordering:*

$$\varphi \leq \psi \Leftrightarrow (\varphi = \psi \vee \varphi = \bot)$$

It is well-known that the category of cpos and continuous functions is cartesian closed and provides a sound and computationally adequate semantics for PCF where the type of natural numbers is interpreted as $\mathbb{N}_\bot$. It is easy to extend this semantics with a type of λ-terms interpreted as $\Lambda_\bot$, a type of lists of λ-terms interpreted as $(\Lambda^*)_\bot$ (where Λ^* denotes the set of finite sequences of λ-terms) and product types interpreted with the categorical product of cpos. All the constants of system ΛT_{bbc} can be interpreted with fixpoints and basic operations on flat domains, therefore the category of cpos and continuous functions forms a model of system ΛT_{bbc}.

We now fix some notations. If $\boldsymbol{\sigma}$ is a type of system ΛT_{bbc}, then $[\![\boldsymbol{\sigma}]\!]$ denotes the cpo interpreting $\boldsymbol{\sigma}$. If a typing derivation in system ΛT_{bbc} has conclusion $x_0 : \boldsymbol{\sigma}_0, \ldots, x_{n-1} : \boldsymbol{\sigma}_{n-1} \vdash \mathtt{M} : \boldsymbol{\tau}$ and if v is a valuation such that $v(x_i) \in [\![\boldsymbol{\sigma}_i]\!]$ for each i, then $[\![\mathtt{M}]\!]_v \in [\![\boldsymbol{\tau}]\!]$ is the denotation of $\mathtt{M}$ with valuation v. The category of cpos and continuous functions provides a sound and computationally adequate model for system ΛT_{bbc}:

Lemma 5. *If $M \rightsquigarrow N$ in system ΛT_{bbc} and if v is a valuation then:*

$$[\![M]\!]_v = [\![N]\!]_v$$

Moreover, if $M : \boldsymbol{\iota}$ is a closed term and if $[\![M]\!]$ is some $\mathfrak{n} \in \mathbb{N}$ then:

$$M \rightsquigarrow^* \mathtt{s}^{\mathfrak{n}}\,\mathtt{z}$$

These results are proved using standard techniques for continuous models of PCF, see e.g. [16]. In system ΛT_{bbc}, computational adequacy in fact holds for every basic type, but in our case we only need it on the type $\boldsymbol{\iota}$ of natural numbers. Finally, we also stress that the BBC functional is a total element in this model, and therefore system ΛT_{bbc} is a total language: all computations terminate. This means that if $\mathtt{M}$ is a closed term in system ΛT_{bbc}, then $[\![\mathtt{M}]\!] \neq \bot$. We do not prove totality of the BBC functional here since the proof is very similar to (and simpler than) its proof of adequacy (lemma 8).

Before ending this section, we mention a result that will be useful for the proof of adequacy of the BBC functional:

Lemma 6. *Write $D^{X_\perp}$ for the domain of continuous functions from $X_\perp$ to D. If φ is a continuous function from $D^{X_\perp}$ to $Y_\perp$ and if $\psi \in D^{X_\perp}$ is such that $\varphi(\psi) \neq \perp$ and $\psi(\perp) = \perp$, then there exists a finite set $F \subseteq X$ such that:*

$$\forall \psi' \in D^{X_\perp} \left(\forall \theta \in F \left(\psi'(\theta) = \psi(\theta)\right) \Rightarrow \varphi(\psi') = \varphi(\psi)\right)$$

Proof. Define for F finite subset of X the continuous function:

$$\psi_F(\theta) = \begin{cases} \psi(\theta) & \text{if } \theta \in F \\ \perp & \text{otherwise} \end{cases}$$

Then $\{\psi_F \mid F \subseteq X \text{ finite}\}$ is directed and:

$$\psi = \sqcup\{\psi_F \mid F \subseteq X \text{ finite}\}$$

so the continuity of φ implies that:

$$\varphi(\psi) = \sqcup\{\varphi(\psi_F) \mid F \subseteq X \text{ finite}\}$$

By definition of the order on $Y_\perp$, this means that there must exist some finite $F \subseteq X$ such that $\varphi(\psi_F) = \varphi(\psi)$. If ψ' is such that $\psi'(\theta) = \psi(\theta)$ for every $\theta \in F$, then $\psi' \geq \psi_F$ so $\varphi(\psi') \geq \varphi(\psi_F) = \varphi(\psi)$. Finally, since $\varphi(\psi) \neq \perp$ we obtain $\varphi(\psi') = \varphi(\psi)$. □

V. Realizability

This section contains the main contribution of our work: the translation of system F into system ΛT_{bbc} through a bar recursive interpretation of second-order arithmetic. Our realizability model follows the lines of Kreisel's modified realizability. A standard Dialectica interpretation for our logic would not be possible because the interpretation of contraction ($A \Rightarrow A \wedge A$) requires the decidability of quantifier-free formulas, which we do not have in our logic ($M \in \mathfrak{X}$ is undecidable when $\mathfrak{X}$ is the set of normalizing terms). However, the Diller-Nahm interpretation [17] overcomes this difficulty and we need to investigate the possibility of using it instead of modified realizability.

We first define a syntactic mapping from our logic into system ΛT_{bbc} and the realizability values of formulas. Then we show how we interpret classical logic, the axiom scheme of comprehension and the instantiation of a set variable with an arbitrary formula. Finally, we define the interpretation of the normalization proof of section II and derive our translation from it.

A. Mapping the logic into system ΛT_{bbc}

In this section we define a mapping $\_^\diamond$ from our logic to system ΛT_{bbc}. We will first map formulas to types, and then elements of computational sort (natural numbers, terms and lists of terms) to programs.

We now map formulas of our logic into types of system ΛT_{bbc}. We define such a mapping because our realizability model is typed, which means that the set of realizers of a formula A will be defined as a subset of the domain interpretation of $A^\diamond$. The mapping of formulas to types is as follows:

$$(A \Rightarrow B)^\diamond = A^\diamond \to B^\diamond \qquad (A \wedge B)^\diamond = A^\diamond \times B^\diamond$$
$$\Phi^\diamond = \boldsymbol{\iota} \qquad (\forall i\, A)^\diamond = \boldsymbol{\iota} \to A^\diamond \qquad (\forall t\, A)^\diamond = \boldsymbol{\lambda} \to A^\diamond$$
$$(\forall \pi\, A)^\diamond = \boldsymbol{\lambda}^* \to A^\diamond \qquad (\forall X\, A)^\diamond = (\forall b\, A)^\diamond = A^\diamond$$

The atomic formulas are mapped to the type of natural numbers. This is because we want to extract natural numbers (bounds on the numbers of reduction steps for reaching a normal form) from proofs in classical logic. Indeed, interpreting atomic formulas with $\boldsymbol{\iota}$ allows us to perform the standard technique of defining the set of realizers of the false formula as a well-chosen subset of the natural numbers. This technique was already used in [6] and is the computational counterpart of Friedman's A-translation. The types universal quantifications are mapped to depend on the sort of the quantified variable. The sorts of natural numbers, terms and lists of terms are called computational: a realizer of a quantification on a computational sort takes an element of that sort as input and builds a realizer of the instantiation of the formula with that element. Conversely, the sorts of sets and booleans are not computational: a realizer of a quantification on a non-computational sort must be uniform, in the sense that it must realize all the instantiations regardless of the element the formula is instantiated with.

Since the type associated to a formula only depends on its propositional structure, the type associated to an instance of a 1-formula $A(\_, \ldots, \_)$ does not depend on the instance and will simply be written $A^\diamond$. On the other hand, the type associated to a 2-formula depends on its particular instance.

Since the realizer of a quantification on a computational sort depends on the element of the logic it is instantiated with, we also need a mapping from elements of computational sort to system ΛT_{bbc} programs of the corresponding type. For simplicity and without loss of generality we suppose that the variables i, t and π of the logic are also variables of system ΛT_{bbc} with respective types $\boldsymbol{\iota}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\lambda}^*$. A first-order element m, M or Π of the logic is then mapped to a program $m^\diamond$, $M^\diamond$ or $\Pi^\diamond$ with the same set of variables. The mapping is defined inductively as follows:

$$i^\diamond = i \qquad 0^\diamond = \mathtt{z} \qquad (S\, m)^\diamond = \mathtt{s}\, m^\diamond$$
$$t^\diamond = t \qquad \underline{m}^\diamond = \mathtt{var}\, m^\diamond \qquad (\lambda.M)^\diamond = \mathtt{abs}\, M^\diamond$$
$$(M\, \Pi)^\diamond = \mathtt{app}^*\, M^\diamond\, \Pi^\diamond \qquad (M\, [\Pi])^\diamond = \mathtt{subst}\, M^\diamond\, \mathtt{z}\, \Pi^\diamond$$
$$\pi^\diamond = \pi \qquad \langle\rangle^\diamond = \mathtt{nil} \qquad \langle \Pi, M \rangle^\diamond = \mathtt{cons}\, \Pi^\diamond\, M^\diamond$$

B. Realizability values

In this section we define the realizability model with which we interpret the normalization proof of section II using programs of system ΛT_{bbc}. The set $|A|$ of realizers of a formula A will be defined as a subset of $[\![A^\diamond]\!]$, where $\_^\diamond$ is the mapping from formulas to types of system ΛT_{bbc} defined in section V-A and $[\![\_]\!]$ is the semantic interpretation of section IV-B.

Because a formula can contain free variables, its realizability value $|\_|$ will depend on a valuation. A valuation v on a formula A is a function v on the free variables of A such that:

$$v(i) \in \mathbb{N} \qquad v(t) \in \Lambda \qquad v(\pi) \in \Lambda^*$$

$$v(X) \in \mathcal{P}(\Lambda) \qquad v(b) \in \{\mathfrak{tt}; \mathfrak{ff}\}$$

where Λ^* denotes the set of finite sequences of λ-terms. Since $\mathbb{N} \subseteq \mathbb{N}_\perp = [\![\boldsymbol{\iota}]\!]$, $\Lambda \subseteq \Lambda_\perp = [\![\boldsymbol{\lambda}]\!]$ and $\Lambda^* \subseteq (\Lambda^*)_\perp = [\![\boldsymbol{\lambda}^*]\!]$, we have that for any term m, M or Π appearing in A, a valuation on A is in particular a valuation in the domain-theoretic sense on $m^\diamond$, $M^\diamond$ or $\Pi^\diamond$, where $\_^\diamond$ is the mapping from computational terms to programs of system ΛT_{bbc} defined in section V-A. Therefore, $[\![m^\diamond]\!]_v \in [\![\boldsymbol{\iota}]\!]$, $[\![M^\diamond]\!]_v \in [\![\boldsymbol{\lambda}]\!]$ and $[\![\Pi^\diamond]\!]_v \in [\![\boldsymbol{\lambda}^*]\!]$ are well-defined. Moreover, $[\![m^\diamond]\!]_v \in \mathbb{N}$, $[\![M^\diamond]\!]_v \in \Lambda$ and $[\![\Pi^\diamond]\!]_v \in \Lambda^*$: they are different from $\perp$.

As explained in the previous section, the set of realizers of false atomic formulas will be a well-chosen set of natural numbers so we can extract computational content from proofs in classical logic. For now this set is a parameter of our realizability model:

$$\perp\!\!\!\perp \subseteq \mathbb{N}$$

From that parameter, we define the realizability value $|A|_v \subseteq [\![A^\diamond]\!]$ of a formula A with valuation v in figure 4. The realizability value of a boolean formula Φ is either the whole set $[\![\Phi^\diamond]\!] = \mathbb{N}_\perp$ or the parameter $\perp\!\!\!\perp$, which is a standard definition in realizability models for classical logic. In the definition of $|M\!\downarrow_\star^m|_v$, remember that $[\![M^\diamond]\!]_v \in \Lambda$ and $[\![m^\diamond]\!]_v \in \mathbb{N}$ (they are not $\perp$), so the definition is correct. The realizability value for quantifications depends on whether the sort of the quantified variable is computational or not. In the case of computational quantifications, the realizers takes as input the element the formula is instantiated with, while for non-computational quantifications the realizer does not depend on the particular value the formula is instantiated with: the realizer is uniform. Realizability values of implication and conjunction are standard.

In the following, we will also use terms and formulas with parameters, instead of valuations. This means that we syntactically substitute elements of $[\![\boldsymbol{\sigma}]\!]$ for free variables of type $\boldsymbol{\sigma}$ in interpretations of terms of system ΛT_{bbc}, and elements of $\mathbb{N}$, Λ, Λ^*, $\mathcal{P}(\Lambda)$ and $\{\mathfrak{tt}; \mathfrak{ff}\}$ for free variables of the corresponding sort in realizability values of formulas. For example, we write $[\![\lambda x.\mathtt{app}\, \varphi\, x]\!]$ instead of $[\![\lambda x.\mathtt{app}\, y\, x]\!]_{\{y \mapsto \varphi\}}$, and we write $\left|\forall t\, t\!\downarrow_\star^{S\ 7}\right|$ for $\left|\forall t\, t\!\downarrow_\star^{S\ i}\right|_{\{i \mapsto 7\}}$. A closed element with parameters is an element with parameters that does not have any free variables anymore.

C. Classical logic

In this section, we explain how we deal with classical logic. As explained in section III, we work in the target of Gödel's negative translation. This means that classical principles can be realized. In particular, we can define realizers of double-negation elimination by induction on formulas:

$$\mathtt{dne}_\Phi = \lambda x.x\,(\lambda y.y) \qquad \mathtt{dne}_{\forall b\, A} = \mathtt{dne}_{\forall X\, A} = \mathtt{dne}_A$$
$$\mathtt{dne}_{\forall \eta\, A} = \lambda x \eta.\mathtt{dne}_A\,(\lambda y.x\,(\lambda z.y\,(z\,\eta)))$$
$$\mathtt{dne}_{A \Rightarrow B} = \lambda x y.\mathtt{dne}_B\,(\lambda z.x\,(\lambda u.z\,(u\,y)))$$
$$\mathtt{dne}_{A \wedge B} =$$
$$\lambda x.\,\langle \mathtt{dne}_A\,(\lambda y.x\,(\lambda z.y\,(\mathrm{p}_1\,z))), \mathtt{dne}_B\,(\lambda y.x\,(\lambda z.y\,(\mathrm{p}_2\,z)))\rangle$$

where η ranges over variables of computational sort i, t and π. These terms indeed realize double-negation elimination:

Lemma 7. *If A is a closed formula with parameters then:*

$$[\![\mathtt{dne}_A]\!] \in |\neg\neg A \Rightarrow A|$$

Proof. By induction on A, the base case $A \equiv \Phi$ is as follows: since by definition $|\Phi|$ is either $|\mathit{tt}|$ or $|\mathit{ff}|$, it is sufficient to check that $[\![\lambda x.x\,(\lambda y.y)]\!] \in |\neg\neg \mathit{tt} \Rightarrow \mathit{tt}|$ and $[\![\lambda x.x\,(\lambda y.y)]\!] \in |\neg\neg \mathit{ff} \Rightarrow \mathit{ff}|$.

- $[\![\lambda x.x\,(\lambda y.y)]\!] \in |\neg\neg \mathit{tt} \Rightarrow \mathit{tt}|$: let $\varphi \in |(\mathit{tt} \Rightarrow \mathit{ff}) \Rightarrow \mathit{ff}|$. We have to show that $[\![\varphi\,(\lambda y.y)]\!] \in |\mathit{tt}|$, but this is immediate since $|\mathit{tt}| = \mathbb{N}_\perp$
- $[\![\lambda x.x\,(\lambda y.y)]\!] \in |\neg\neg \mathit{ff} \Rightarrow \mathit{ff}|$: let $\varphi \in |(\mathit{ff} \Rightarrow \mathit{ff}) \Rightarrow \mathit{ff}|$. We have to show that $[\![\varphi\,(\lambda y.y)]\!] \in |\mathit{ff}|$, which is true because $[\![\lambda y.y]\!] \in |\mathit{ff} \Rightarrow \mathit{ff}|$ □

We also define the following term:

$$\mathtt{exf}_A = \lambda x.\mathtt{dne}_A\,(\lambda \_.x)$$

which immediately realizes the *ex falso quodlibet* principle:

$$[\![\mathtt{exf}_A]\!] \in |\mathit{ff} \Rightarrow A|$$

D. Realizing the axiom scheme of comprehension

It is well known that the combination of the axiom of countable choice with classical logic implies the comprehension scheme on natural numbers. The idea is that classical logic provides a proof of $\forall i\, \exists b\ (b \Leftrightarrow A(i))$. Then using the axiom of choice, we obtain $\exists f\, \forall i\ (f(i) \Leftrightarrow A(i))$, where f is a function from natural numbers to booleans. Therefore, we can interpret second-order arithmetic through an encoding of sets of natural numbers as functions from natural numbers to booleans.

In our case, we want to interpret the comprehension scheme on λ-terms rather than on natural numbers. Therefore, we will interpret the following version of the axiom of countable choice:

$$\forall t\, \exists b\, A(b,t) \Rightarrow \exists X\, \forall t\, A(t \in X, t)$$

We actually interpret a slightly weaker version: we define a program that turns an element of $\bigcap_{\mathfrak{M} \in \Lambda} |\exists b\, A(b, \mathfrak{M})|$ into an element of $|\exists X\, \forall t\, A(t \in X, t)|$. The difference is that a realizer of $\forall t\, \exists b\, A(b,t)$ takes a term as input (since the sort of terms is computational), while in our particular case we can build a realizer of $\exists b\ (b \Leftrightarrow A(t))$ that is uniform in t. Because of that, the weaker version is sufficient for interpreting the comprehension scheme. The usual BBC functional [6] (where the first argument would be of type $\boldsymbol{\lambda} \to (\boldsymbol{\sigma} \to \boldsymbol{\iota}) \to \boldsymbol{\sigma}$) can in fact realize the stronger version where the left quantification on t is relativized. Our version is slightly weaker because the first argument is only of type $(\boldsymbol{\sigma} \to \boldsymbol{\iota}) \to \boldsymbol{\sigma}$. It is not clear yet whether the usual versions are computationally strictly stronger than our version. Our proof of adequacy is inspired by [7] and uses Zorn's lemma:

Lemma 8. *If $A(\Phi, M)$ is a closed 1-formula with parameters and if $\varphi \in \bigcap_{\mathfrak{M} \in \Lambda} |\exists b\, A(b, \mathfrak{M})|$ then:*

$$[\![\lambda x.bbc\,(\lambda y.\mathtt{exf}_A\,(\varphi\, y))\, x\, \{\}]\!] \in |\exists X\, \forall t\, A(t \in X, t)|$$

$$|b|_v = \begin{cases} \mathbb{N}_\perp & \text{if } v(b) = \mathtt{tt} \\ \perp\!\!\!\perp & \text{if } v(b) = \mathtt{ff} \end{cases} \qquad |\mathtt{tt}|_v = \mathbb{N}_\perp \qquad |\mathtt{ff}|_v = \perp\!\!\!\perp \qquad |M \in X|_v = \begin{cases} \mathbb{N}_\perp & \text{if } [\![M^\diamond]\!]_v \in v(X) \\ \perp\!\!\!\perp & \text{if } [\![M^\diamond]\!]_v \notin v(X) \end{cases}$$

$$|M\not\searrow^m|_v = \begin{cases} \mathbb{N}_\perp & \text{if } [\![M^\diamond]\!]_v \text{ can reduce for } [\![m^\diamond]\!]_v \text{ steps of weak head reduction without reaching a normal form} \\ \perp\!\!\!\perp & \text{otherwise} \end{cases}$$

$$|A \Rightarrow B|_v = \{\varphi \in [\![A^\diamond \to B^\diamond]\!] \mid \forall \psi \in |A|_v, \varphi(\psi) \in |B|_v\} \qquad |A \wedge B|_v = \{(\varphi,\psi) \in [\![A^\diamond \times B^\diamond]\!] \mid \varphi \in |A|_v \wedge \psi \in |B|_v\}$$

$$|\forall i\, A|_v = \left\{\varphi \in [\![\boldsymbol{\iota} \to A^\diamond]\!] \;\middle|\; \forall \mathfrak{n} \in \mathbb{N}, \varphi(\mathfrak{n}) \in |A|_{v \uplus \{i \mapsto \mathfrak{n}\}}\right\} \qquad |\forall t\, A|_v = \left\{\varphi \in [\![\boldsymbol{\lambda} \to A^\diamond]\!] \;\middle|\; \forall \mathfrak{M} \in \Lambda, \varphi(\mathfrak{M}) \in |A|_{v \uplus \{t \mapsto \mathfrak{M}\}}\right\}$$

$$|\forall \pi\, A|_v = \left\{\varphi \in [\![\boldsymbol{\lambda}^* \to A^\diamond]\!] \;\middle|\; \forall \mathfrak{p} \in \Lambda^*, \varphi(\mathfrak{p}) \in |A|_{v \uplus \{\pi \mapsto \mathfrak{p}\}}\right\}$$

$$|\forall X\, A|_v = \bigcap_{\mathfrak{X} \in \mathcal{P}(\Lambda)} |A|_{v \uplus \{X \mapsto \mathfrak{X}\}} \qquad |\forall b\, A|_v = \bigcap_{\mathfrak{b} \in \{\mathtt{tt};\mathtt{ff}\}} |A|_{v \uplus \{b \mapsto \mathfrak{b}\}}$$

Fig. 4. Realizability values

Proof. Let $\psi \in |\forall X\, \neg\forall t\, A(t \in X, t)|$ and write $\theta = [\![\mathtt{bbc}\,(\lambda y.\mathtt{exf}_A\,(\varphi\, y))\,\psi]\!]$. We have to prove that:

$$[\![\theta\,\{\}]\!] \in |\mathit{ff}|$$

First, we fix the set E of $\xi \in \left[\!\left[A^{\diamond\dagger}\right]\!\right]$ such that:

- $\pi_2(\xi(\mathfrak{M})) \in |A(\mathtt{tt},\mathfrak{M})| \cup |A(\mathtt{ff},\mathfrak{M})|$ if $\pi_1(\xi(\mathfrak{M})) = 0$
- $\xi(\mathfrak{M}) = (1, [\![\mathtt{can}_{A^\diamond}]\!])$ otherwise
- $\xi(\perp) = \perp$
- $\theta(\xi) \notin |\mathit{ff}|$

and we define a partial order $\prec$ on E by:

$$\xi \prec \xi' \iff (\pi_1(\xi(\mathfrak{M})) = 0 \Rightarrow \xi'(\mathfrak{M}) = \xi(\mathfrak{M}))$$

We will prove that every non-empty chain of E has an upper bound in E and that E has no maximal element. Therefore by Zorn's lemma the empty set cannot have an upper bound in E and so $E = \emptyset$. In particular $[\![\{\}]\!] \notin E$ and so $[\![\theta\,\{\}]\!] = \theta([\![\{\}]\!]) \in |\mathit{ff}|$ because $[\![\{\}]\!]$ satisfies all other conditions of E (since $\{\}$ is strict).

- Every non-empty chain of E has an upper bound in E:
  Let C be a non-empty chain of E and build ξ_{max} as follows:

  $$\xi_{max}(\mathfrak{M}) = \begin{cases} \xi(\mathfrak{M}) & \text{if } \pi_1(\xi(\mathfrak{M})) = 0 \text{ for some } \xi \in C \\ (1, [\![\mathtt{can}_{A^\diamond}]\!]) & \text{otherwise} \end{cases}$$

  $$\xi_{max}(\perp) = \perp$$

  This function is well-defined because C is a chain for $\prec$ so if $\xi, \xi' \in C$ are such that $\pi_1(\xi(\mathfrak{M})) = \pi_1(\xi'(\mathfrak{M})) = 0$ for some $\mathfrak{M}$, then $\xi(\mathfrak{M}) = \xi'(\mathfrak{M})$. Also, if $\pi_1(\xi_{max}(\mathfrak{M})) = 0$ then $\pi_1(\xi(\mathfrak{M})) = 0$ for some $\xi \in C$, and therefore:

  $$\pi_2(\xi_{max}(\mathfrak{M})) = \pi_2(\xi(\mathfrak{M})) \in |A(\mathtt{tt},\mathfrak{M})| \cup |A(\mathtt{ff},\mathfrak{M})|$$

  The only non-trivial property left to prove in order to get $\xi_{max} \in E$ is that $\theta(\xi_{max}) \notin |\mathit{ff}|$. Suppose $\theta(\xi_{max}) \in |\mathit{ff}|$. Then, $\theta(\xi_{max}) \neq \perp$ because $|\mathit{ff}| = \perp\!\!\!\perp \subseteq \mathbb{N}$. Moreover, $\xi_{max}(\perp) = \perp$ so we can apply lemma 6 with $X = \Lambda$, $D = [\![\boldsymbol{\iota} \times A^\diamond]\!]$ and $Y = \mathbb{N}$ to get a finite set $F \subseteq \Lambda$ such that:

  $$\forall \xi\, (\forall \mathfrak{M} \in F\, (\xi(\mathfrak{M}) = \xi_{max}(\mathfrak{M})) \Rightarrow \theta(\xi) = \theta(\xi_{max}))$$

  For every $\mathfrak{M} \in F$ there exists some $\xi_{\mathfrak{M}} \in C$ such that $\xi_{\mathfrak{M}}(\mathfrak{M}) = \xi_{max}(\mathfrak{M})$: if $\pi_1(\xi_{max}(\mathfrak{M})) = 0$ then this is by definition of ξ_{max} and if $\pi_1(\xi_{max}(\mathfrak{M})) \neq 0$ then any element of C meets the condition (remember that C is non-empty). C is a non-empty chain and $\{\xi_{\mathfrak{M}} \mid \mathfrak{M} \in F\}$ is a finite subset of C so it has a greatest element $\xi_{\mathfrak{M}_0}$. Then it is easy to see that for any $\mathfrak{M} \in F$, $\xi_{\mathfrak{M}_0}(\mathfrak{M}) = \xi_{max}(\mathfrak{M})$. Therefore $\theta(\xi_{max}) = \theta(\xi_{\mathfrak{M}_0}) \notin |\mathit{ff}|$ since $\xi_{\mathfrak{M}_0} \in C \subseteq E$, hence the contradiction.
- E has no maximal element:
  Suppose for the sake of contradiction that ξ is some maximal element of E. We have the following equation:

  $$[\![\theta\,\xi]\!] = [\![\psi\,(\xi \mid \lambda y.\mathtt{exf}_A\,(\varphi\,(\lambda z.\theta\,(\xi \cup \{y \mapsto z\}))))]\!]$$

  Let $\mathfrak{X} = \{\mathfrak{M} \in \Lambda \mid \pi_2(\xi(\mathfrak{M})) \in |A(\mathtt{tt},\mathfrak{M})|\}$. Since we have $\psi \in |\neg\forall t\, A(t \in \mathfrak{X}, t)|$ and $[\![\theta\,\xi]\!] = \theta(\xi) \notin |\mathit{ff}|$, we get:

  $$[\![\xi \mid \lambda y.\mathtt{exf}_A\,(\varphi\,(\lambda z.\theta\,(\xi \cup \{y \mapsto z\})))]\!] \notin |\forall t\, A(t \in \mathfrak{X}, t)|$$

  Therefore there is some $\mathfrak{M} \in \Lambda$ such that:

  $$[\![(\xi \mid \lambda y.\mathtt{exf}_A\,(\varphi\,(\lambda z.\theta\,(\xi \cup \{y \mapsto z\}))))\,\mathfrak{M}]\!] \notin |A(\mathfrak{M} \in \mathfrak{X}, \mathfrak{M})|$$

  If $\pi_1(\xi(\mathfrak{M})) = 0$ then $\pi_2(\xi(\mathfrak{M})) \notin |A(\mathfrak{M} \in \mathfrak{X}, \mathfrak{M})|$, but since $\xi \in E$ we also have:

  $$\pi_2(\xi(\mathfrak{M})) \in |A(\mathtt{tt},\mathfrak{M})| \cup |A(\mathtt{ff},\mathfrak{M})|$$

  and we have a contradiction by definition of $\mathfrak{X}$. Therefore $\pi_1(\xi(\mathfrak{M})) \neq 0$. Moreover $\pi_1(\xi(\mathfrak{M})) \neq \perp$ because $\xi \in E$, so we obtain:

  $$[\![\mathtt{exf}_A\,(\varphi\,(\lambda z.\theta\,(\xi \cup \{\mathfrak{M} \mapsto z\})))]\!] \notin |A(\mathfrak{M} \in \mathfrak{X}, \mathfrak{M})|$$

  and therefore $[\![\varphi\,(\lambda z.\theta\,(\xi \cup \{\mathfrak{M} \mapsto z\}))]\!] \notin |\mathit{ff}|$. Finally, since $\varphi \in |\neg\forall b\, \neg A(b,\mathfrak{M})|$, we have:

  $$[\![\lambda z.\theta\,(\xi \cup \{\mathfrak{M} \mapsto z\})]\!] \notin |\forall b\, \neg A(b,\mathfrak{M})|$$

  which means that there exists some:

  $$\zeta \in |A(\mathtt{tt},\mathfrak{M})| \cup |A(\mathtt{ff},\mathfrak{M})|$$

such that $[\![\theta(\xi \cup \{\mathfrak{M} \mapsto \zeta\})]\!] \notin |ff|$. It is then easy to check that $[\![\xi \cup \{\mathfrak{M} \mapsto \zeta\}]\!] \in E$ and $\xi \prec [\![\xi \cup \{\mathfrak{M} \mapsto \zeta\}]\!]$, contradicting the maximality of ξ. □

As explained before the lemma, the next step is the definition of an element of $\bigcap_{\mathfrak{M}\in\Lambda} |\exists b\,(b \Leftrightarrow A(\mathfrak{M}))|$, so that its combination with the realizer above provides an interpretation of the comprehension scheme: $\exists X\,\forall t\,(t \in X \Leftrightarrow A(t))$.

Lemma 9. *If $A(M)$ is a closed 1-formula with parameters such that $b \notin FV(A(t))$, then:*

$$[\![\lambda x.x\,\langle \mathtt{exf}_A, \lambda y.x\,\langle \lambda\_.y, \lambda\_.z\rangle\rangle]\!] \in \bigcap_{\mathfrak{M}\in\Lambda} |\exists b\,(b \Leftrightarrow A(\mathfrak{M}))|$$

Proof. Let $\mathfrak{M} \in \Lambda$ and $\varphi \in |\forall b \neg (b \Leftrightarrow A(\mathfrak{M}))|$. We have to prove that:

$$[\![\varphi\,\langle \mathtt{exf}_A, \lambda y.\varphi\,\langle \lambda\_.y, \lambda\_.\mathrm{z}\rangle\rangle]\!] \in |ff|$$

Since $\varphi \in |\neg(\mathbb{ff} \Leftrightarrow A(\mathfrak{M}))|$, it is sufficient to prove:

$$[\![\mathtt{exf}_A]\!] \in |\mathbb{ff} \Rightarrow A(\mathfrak{M})|$$
$$[\![\lambda y.\varphi\,\langle \lambda\_.y, \lambda\_.\mathrm{z}\rangle]\!] \in |\neg A(\mathfrak{M})|$$

The first one is immediate. For the second, let $\psi \in |A(\mathfrak{M})|$. Since $\varphi \in |\neg(\mathbb{t} \Leftrightarrow A(\mathfrak{M}))|$, it is sufficient to prove:

$$[\![\lambda\_.\psi]\!] \in |\mathbb{t} \Rightarrow A(\mathfrak{M})|$$
$$[\![\lambda\_.\mathrm{z}]\!] \in |A(\mathfrak{M}) \Rightarrow \mathbb{t}|$$

The first one is immediate, and the second one follows from the fact that $|\mathbb{t}| = \mathbb{N}_\perp$. □

Combining the two realizers above, we can now define:

$$\mathtt{comp}_A = \lambda x.\mathrm{bbc}\,(\lambda y.\mathtt{exf}_A\,(y\,\langle \mathtt{exf}_A, \lambda u.y\,\langle \lambda\_.u, \lambda\_.\mathrm{z}\rangle\rangle))\,x\,\{\}$$

which by construction realizes the axiom scheme of comprehension:

$$[\![\mathtt{comp}_A]\!] \in |\exists X\,\forall t\,(t \in X \Leftrightarrow A(t))|$$

E. Realizing second-order elimination

We have now realized the existence of a first-order element of sort set witnessing any formula. However, we still need to subtitute an arbitrary 1-formula for a first-order set variable. In other words, we have to interpret:

$$\forall X A\left(\overline{X}\right) \Rightarrow A(B)$$

for arbitrary 2-formula $A(C)$ and 1-formula $B(M)$. The first step towards the interpretation of second-order elimination is the interpretation of the following formula:

$$\forall t\,(B(t) \Leftrightarrow C(t)) \Rightarrow (A(B) \Leftrightarrow A(C))$$

The combination of a realizer of that formula with $\mathtt{comp}_B$ will then allow us to deduce $A(B)$ from $A\left(\overline{X}\right)$.

Since we build the realizer $\mathtt{repl}_A$ of that formula by induction on A, we need to explicitly take into account the free variables of A which are of computational sort. That means that $\mathtt{repl}_A$ will be such that $\mathrm{FV}(\mathtt{repl}_A) = \mathrm{FV}(A) \cap \eta$ (by $\mathrm{FV}(A) \cap \eta$ we mean the free variables of A that are of a computational sort, i.e. i, t or π). In particular, if A is closed then $\mathtt{repl}_A$ is closed as well. For simplicity, we will actually define $\mathtt{repl}'_A$ such that $\mathrm{FV}(\mathtt{repl}'_A) = \mathrm{FV}(\mathtt{repl}_A) \cup \{x\}$, and then define $\mathtt{repl}_A = \lambda x.\mathtt{repl}'_A$. The definition of $\mathtt{repl}'_A$ is given in figure 5. We can then prove the intended result by induction on A:

Lemma 10. *If $A(D)$ is a 2-formula, $B(M)$, $C(M)$ are closed 1-formulas with parameters and v is a valuation on A then:*

$$[\![\mathtt{repl}_A]\!]_v \in |\forall t\,(B(t) \Leftrightarrow C(t)) \Rightarrow (A(B) \Leftrightarrow A(C))|_v$$

We now have all the ingredients to interpret the instantiation of a set variable with an arbitrary 1-formula $B(M)$:

$$\forall X\,A\left(\overline{X}\right) \Rightarrow A(B)$$

Since the existential quantifier is not primitive in our logic, our version of the axiom scheme of comprehension is in fact:

$$\neg\forall X\,\neg\forall t\,(t \in X \Leftrightarrow B(t))$$

therefore, the elimination of such an existential quantifier will require classical logic. Our realizer $\mathtt{elim}_{A,B}$ of second-order elimination (where $A(C)$ is a 2-formula and $B(M)$ is a 1-formula) is such that $\mathrm{FV}(\mathtt{elim}_{A,B}) = \mathrm{FV}(A) \cap \eta$ and is defined as:

$$\mathtt{elim}_{A,B} = \lambda x.\mathtt{dne}_{A(B)}\,(\lambda y.\mathtt{comp}_B\,(\lambda z.y\,(\mathrm{p}_1\,(\mathtt{repl}_A\,z)\,x)))$$

Correctness of this realizer is then an easy consequence of the lemmas above:

Lemma 11. *If $A(C)$ is a 2-formula such that $X \notin FV(A)$ (meaning that $X \notin FV(A(C))$ whenever $X \notin FV(C)$), if $B(M)$ is a closed 1-formula with parameters and if v is a valuation on A then:*

$$[\![\mathtt{elim}_{A,B}]\!]_v \in \left|\forall X\,A\left(\overline{X}\right) \Rightarrow A(B)\right|_v$$

Proof. Let $\varphi \in \left|\forall X\,A\left(\overline{X}\right)\right|_v$. Since:

$$\left[\!\left[\mathtt{dne}_{A(B)}\right]\!\right] \in |\neg\neg A(B) \Rightarrow A(B)|_v$$

we are left to prove:

$$[\![\lambda y.\mathtt{comp}_B\,(\lambda z.y\,(\mathrm{p}_1\,(\mathtt{repl}_A\,z)\,\varphi))]\!]_v \in |\neg\neg A(B)|_v$$

Let $\psi \in |\neg A(B)|_v$. Since:

$$[\![\mathtt{comp}_B]\!] \in |\neg\forall X\,\neg\forall t\,(t \in X \Leftrightarrow B(t))|$$

we are left to prove:

$$[\![\lambda z.\psi\,(\mathrm{p}_1\,(\mathtt{repl}_A\,z)\,\varphi)]\!]_v \in |\forall X\,\neg\forall t\,(t \in X \Leftrightarrow B(t))|_v$$

Let $\mathfrak{X} \subseteq \Lambda$ and $\theta \in |\forall t\,(t \in \mathfrak{X} \Leftrightarrow B(t))|_v$. We need to prove:

$$[\![\psi\,(\mathrm{p}_1\,(\mathtt{repl}_A\,\theta)\,\varphi)]\!]_v \in |ff|_v$$

but we have:

$$[\![\mathrm{p}_1\,(\mathtt{repl}_A\,\theta)]\!]_v \in \left|A\left(\overline{\mathfrak{X}}\right) \Rightarrow A(B)\right|_v$$

and since $\varphi \in \left|A\left(\overline{\mathfrak{X}}\right)\right|_v$ we get:

$$[\![\mathrm{p}_1\,(\mathtt{repl}_A\,\theta)\,\varphi]\!]_v \in |A(B)|_v$$

finally, since $\psi \in |\neg A(B)|_v$ we obtain:

$$[\![\psi\,(\mathrm{p}_1\,(\mathtt{repl}_A\,\theta)\,\varphi)]\!]_v \in |ff|_v$$

□

$$\mathtt{repl}'_{\overline{X}\mapsto M\in X} = x\,M^{\diamond} \qquad \mathtt{repl}'_{\overline{X}\mapsto\Phi} = \langle \lambda y.y, \lambda y.y\rangle \text{ if } \Phi \not\equiv M \in X$$
$$\mathtt{repl}'_{A_1\Rightarrow A_2} = \left\langle \lambda yz.\mathrm{p}_1\,\mathtt{repl}'_{A_2}\left(y\left(\mathrm{p}_2\,\mathtt{repl}'_{A_1}\,z\right)\right), \lambda yz.\mathrm{p}_2\,\mathtt{repl}'_{A_2}\left(y\left(\mathrm{p}_1\,\mathtt{repl}'_{A_1}\,z\right)\right)\right\rangle$$
$$\mathtt{repl}'_{A_1\wedge A_2} = \left\langle \lambda y.\left\langle \mathrm{p}_1\,\mathtt{repl}'_{A_1}\left(\mathrm{p}_1\,y\right), \mathrm{p}_1\,\mathtt{repl}'_{A_2}\left(\mathrm{p}_2\,y\right)\right\rangle, \lambda y.\left\langle \mathrm{p}_2\,\mathtt{repl}'_{A_1}\left(\mathrm{p}_1\,y\right), \mathrm{p}_2\,\mathtt{repl}'_{A_2}\left(\mathrm{p}_2\,y\right)\right\rangle\right\rangle$$
$$\mathtt{repl}'_{\forall\eta\,A} = \langle \lambda y\eta.\mathrm{p}_1\,\mathtt{repl}'_A\left(y\,\eta\right), \lambda y\eta.\mathrm{p}_2\,\mathtt{repl}'_A\left(y\,\eta\right)\rangle \qquad \mathtt{repl}'_{\forall X\,A} = \mathtt{repl}'_{\forall b\,A} = \mathtt{repl}'_A$$

Fig. 5. Definition of $\mathtt{repl}_A$

F. Realizing normalization of system F

We now have an interpretation of full second-order arithmetic. Therefore, we can describe the details of our interpretation of the proof of normalization of system F given in section II using the realizer `elim` of previous section. The first realizer corresponds to lemma 2 and is defined as follows:

$$\mathtt{normrc} = \left\langle \left\langle \mathtt{normrc}^{(1)}, \mathtt{normrc}^{(2)} \right\rangle, \mathtt{normrc}^{(3)} \right\rangle$$

where:

$$\mathtt{normrc}^{(1)} = \lambda\pi x.x\,\mathrm{z} \qquad \mathtt{normrc}^{(2)} = \lambda tx.x$$
$$\mathtt{normrc}^{(3)} = \lambda tu\pi xy.x\left(\lambda i.y\left(\mathrm{s}\,i\right)\right)$$

`normrc` can be shown to be the computational interpretation of the proof of lemma 2:

Lemma 12.

$$[\![\mathtt{normrc}]\!] \in |\mathcal{R}edC\!and\left(\Downarrow\right)|$$

In the next step, we give the interpretation of the proof of lemma 3: if we have $\mathcal{R}edC\!and\left(\overline{X}\right)$ for each $X \in \mathrm{FV}\left(T\right)$, then we have $\mathcal{R}edC\!and\left(RC_T\right)$. For that we inductively define in figure 6 for each type T of system F built from variables X of the logic a term:

$$\mathtt{isrc}_T = \left\langle \left\langle \mathtt{isrc}_T^{(1)}, \mathtt{isrc}_T^{(2)} \right\rangle, \mathtt{isrc}_T^{(3)} \right\rangle$$

such that $\mathrm{FV}\left(\mathtt{isrc}_T\right) = \{x_X \mid X \in \mathrm{FV}\left(T\right)\}$. Our claim is then that if we substitute a realizer of $\mathcal{R}edC\!and\left(\overline{X}\right)$ for each corresponding variable x_X, we obtain a realizer of $\mathcal{R}edC\!and\left(RC_T\right)$:

Lemma 13. *If T is a type of system F built from variables X of the logic, if $v : FV(T) \to \mathcal{P}(\Lambda)$ is a valuation on $\mathcal{R}edC\!and\left(RC_T\right)$ and if $v' : \{x_X \mid X \in FV(T)\} \to \left[\!\left[\mathcal{R}edC\!and\left(\overline{X}\right)^{\diamond}\right]\!\right]$ (this codomain does not depend on the particular X chosen) is a valuation such that $v'\left(x_X\right) \in \left|\mathcal{R}edC\!and\left(\overline{X}\right)\right|_v$ for each $X \in FV(T)$, then:*

$$[\![\mathtt{isrc}_T]\!]_{v'} \in \left|\mathcal{R}edC\!and\left(RC_T\right)\right|_v$$

The last step of the interpretation of normalization of system F is the interpretation of lemma 4, which is given in figure 7. Despite the fact that each term defined there depends on a full typing derivation in system F, we use the informal notation $\mathtt{adeq}_{\Gamma\vdash M:T}$, refering to the full derivation only by its conclusion. In order to ease our definition, the terms $\mathtt{adeq}_{\Gamma\vdash M:T}$ contain the following free variables:

$$\{x_X \mid X \in \mathrm{FV}\left(\Gamma, T\right)\} \cup \{t_U \mid U \in \Gamma\} \cup \{y_U \mid U \in \Gamma\}$$

where x_X is meant to be replaced with a realizer of $\mathcal{R}edC\!and\left(\overline{X}\right)$, t_U is meant to be replaced with some term $\mathfrak{M}_U \in \Lambda$ and y_U is meant to be replaced with a realizer of $RC_U\left(\mathfrak{M}_U\right)$. In the notations t_U and y_U, U refers to an occurence of U in Γ, rather than to U itself. The notation $\vec{t_\Gamma}$ in figure 7 and in the lemma is a shorthand for $\mathtt{cons}\left(\mathtt{cons}\left(\ldots\mathtt{cons}\,\mathtt{nil}\,t_{U_0}\ldots\right)t_{U_{n-1}}\right)$ if $\Gamma = U_{n-1},\ldots,U_0$. It can then be shown that the terms $\mathtt{adeq}_{\Gamma\vdash M:T}$ satisfy the intended property:

Theorem 2. *If $\Gamma \vdash M : T$ is a valid typing judgement in system F, and if v is a valuation such that:*

- *$v\left(X\right) \subseteq \Lambda$ and $v\left(x_X\right) \in \left|\mathcal{R}edC\!and\left(\overline{X}\right)\right|_v$ for each variable $X \in FV\left(\Gamma, T\right)$*
- *$v\left(t_U\right) \in \Lambda$ and $v\left(y_U\right) \in \left|RC_U\left(t_U\right)\right|_v$ for each $U \in \Gamma$*

then:

$$[\![\mathtt{adeq}_{\Gamma\vdash M:T}]\!]_v \in \left|RC_T\left(M\left[\vec{t_\Gamma}\right]\right)\right|_v$$

Finally, if a closed term M is of closed type T in system F we can define:

$$\mathtt{norm}_{\vdash M:T} = \mathtt{isrc}_T^{(2)}\,M^{\diamond}\,\mathtt{adeq}_{\vdash M:T}$$

Immediately, we have:

$$[\![\mathtt{norm}_{\vdash M:T}]\!] \in |M{\downarrow}| = \left|\neg\forall i\,M{\downarrow}^i\right|$$

The final step is the extraction of a witness $\mathfrak{n} \in \mathbb{N}$ such that M normalizes in at most $\mathfrak{n}$ steps of weak head reduction. The technique is standard in realizability for classical logic and requires that we fix the set of realizers of false boolean formulas to a well-chosen set:

Theorem 3. *If a closed term M is of closed type T in system F, then $\mathtt{norm}_{\vdash M:T}\left(\lambda x.x\right)$ reduces to some $s^{\mathfrak{n}}\,z$ where $\mathfrak{n}$ is such that M reaches a weak head normal form in at most $\mathfrak{n}$ steps.*

Proof. We first fix the set of realizers of false boolean formulas:

$$\perp\!\!\!\perp = \{\mathfrak{n} \in \mathbb{N} \mid M \text{ reaches a normal form in at most } \mathfrak{n} \text{ steps}\}$$

Now we prove that:

$$[\![\lambda x.x]\!] \in \left|\forall i\,M{\downarrow}^i\right|$$

Indeed, let $\mathfrak{n} \in \mathbb{N}$ and let show that $\mathfrak{n} \in |M{\downarrow}^{\mathfrak{n}}|$. If $\mathfrak{n} \in \perp\!\!\!\perp$ then $[\![M^{\diamond}]\!] = M$ reaches a normal form in at most $\mathfrak{n}$ steps so $|M{\downarrow}^{\mathfrak{n}}| = \perp\!\!\!\perp$ and therefore $\mathfrak{n} \in |M{\downarrow}^{\mathfrak{n}}|$. If $\mathfrak{n} \notin \perp\!\!\!\perp$ then $[\![M^{\diamond}]\!] = M$ can reduce for $\mathfrak{n}$ steps without reaching a normal form so $|M{\downarrow}^{\mathfrak{n}}| = \mathbb{N}_{\perp}$ and therefore $\mathfrak{n} \in |M{\downarrow}^{\mathfrak{n}}|$ trivially. Using that result, we obtain:

$$[\![\mathtt{norm}_{\vdash M:T}\left(\lambda x.x\right)]\!] \in |\mathit{ff}| = \perp\!\!\!\perp$$

$$\mathtt{isrc}_X^{(1)} = \mathrm{p}_1\,(\mathrm{p}_1\,x_X) \qquad \mathtt{isrc}_X^{(2)} = \mathrm{p}_2\,(\mathrm{p}_1\,x_X) \qquad \mathtt{isrc}_X^{(3)} = \mathrm{p}_2\,x_X \qquad \mathtt{isrc}_{T\to U}^{(1)} = \lambda\pi tx.\mathtt{isrc}_U^{(1)}\,(\mathtt{cons}\,\pi\,t)$$
$$\mathtt{isrc}_{T\to U}^{(2)} = \lambda tx.\mathtt{isrc}_U^{(2)}\,(\mathtt{app}\,t\,(\mathtt{var}\,\mathtt{z}))\left(x\,(\mathtt{var}\,\mathtt{z})\left(\mathtt{isrc}_T^{(1)}\,\mathtt{nil}\right)\right) \qquad \mathtt{isrc}_{T\to U}^{(3)} = \lambda tu\pi xvy.\mathtt{isrc}_U^{(3)}\,t\,u\,(\mathtt{cons}\,\pi\,v)\,(x\,v\,y)$$
$$\mathtt{isrc}_{\forall X\,T}^{(1)} = \lambda\pi x_X.\mathtt{isrc}_T^{(1)}\pi \qquad \mathtt{isrc}_{\forall X\,T}^{(3)} = \lambda tu\pi yx_X.\mathtt{isrc}_T^{(3)}\,t\,u\,\pi\,(y\,x_X)$$
$$\mathtt{isrc}_{\forall X\,T}^{(2)} = \lambda tx.\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}edCand(\overline{X})\Rightarrow\forall t(RC_T(t)\Rightarrow\Downarrow),\Downarrow}\left(\lambda x_X\mathtt{isrc}_T^{(2)}\right)\mathtt{normrc}\,t\left(\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}edCand(\overline{X})\Rightarrow RC_T(t),\Downarrow}\,x\,\mathtt{normrc}\right)$$

Fig. 6. Definition of $\mathtt{isrc}_T$

$$\mathtt{adeq}_{\Gamma\vdash\underline{m}:U} = y_U \qquad \mathtt{adeq}_{\Gamma\vdash\lambda.M:U\to T} = \lambda t_U y_U.\mathtt{isrc}_T^{(3)}\left(\mathtt{subst}\,M^\diamond\,(\mathtt{s}\,\mathtt{z})\left(\mathtt{shift}^*\,\vec{t_\Gamma}\right)\right)t_U\,\mathtt{nil}\,\mathtt{adeq}_{\Gamma,U\vdash M:T}$$
$$\mathtt{adeq}_{\Gamma\vdash M\,N:T} = \mathtt{adeq}_{\Gamma\vdash M:U\to T}\left(\mathtt{subst}\,N^\diamond\,\mathtt{z}\,\vec{t_\Gamma}\right)\mathtt{adeq}_{\Gamma\vdash N:U} \qquad \mathtt{adeq}_{\Gamma\vdash M:\forall X\,T} = \lambda x_X.\mathtt{adeq}_{\Gamma\vdash M:T}$$
$$\mathtt{adeq}_{\Gamma\vdash M:T\{U/X\}} = \mathtt{elim}_{\overline{X}\mapsto\mathcal{R}edCand(\overline{X})\Rightarrow RC_T\left(M\left[\vec{t_\Gamma}\right]\right),RC_U}\,\mathtt{adeq}_{\Gamma\vdash M:\forall X\,T}\,\mathtt{isrc}_U$$

Fig. 7. Definition of $\mathtt{adeq}_{\Gamma\vdash M:T}$

Now, computational adequacy of the model with respect to system ΛT_{bbc} implies that $\mathtt{norm}_{\vdash M:T}\,(\lambda x.x)$ reduces to some $\mathtt{s}^{\mathfrak{n}}\mathtt{z}$ where $\mathfrak{n}\in\mathbb{\perp\!\!\!\perp}$, so $\mathfrak{n}$ is such that M reaches a weak head normal form in at most $\mathfrak{n}$ steps. □

It is easy to implement one-step weak head reduction in system ΛT_{bbc}, that is, there exists a term $\mathtt{red} : \boldsymbol{\lambda}\to\boldsymbol{\lambda}$ such that for every λ-term M:

- if $M \succ N$, then $\mathtt{red}\,M^\diamond \rightsquigarrow^* N^\diamond$
- if M is in weak head normal form then $\mathtt{red}\,M^\diamond \rightsquigarrow^* M^\diamond$

Therefore, using our extracted bound we can compute the normal form of any closed λ-term M of closed type T in system F:

$$\mathtt{it}_\iota\,M^\diamond\,\mathtt{red}\,(\mathtt{norm}_{\vdash M:T}\,(\lambda x.x)) \rightsquigarrow \mathtt{U}$$

where $\mathtt{U}$ is the representation of the normal form of M in system ΛT_{bbc}.

VI. Future works

The main difficulty of our translation is the interpretation of the axiom scheme of comprehension, and then the instantiation of a first-order set variable with an arbitrary formula. This part is rather canonical and difficult to improve but the other components could be modified in many ways to provide a more direct translation in the future.

One could extract the normal form directly, rather than a bound on the number of reduction steps. The main issue is the computation of the normal form of M from that of $M\,\underline{0}$. To do so we could consider not only weak head reduction, but full head reduction. One could also interpret of a proof of strong normalization. Doing so would require complicated realizers, but provide a way to obtain β-normal forms rather than (weak) head normal forms.

Implementing the translation can be done using deep or shallow embedding. Since the realizers depend on the types appearing in the typing derivation of system F, shallow embedding requires an untyped or a dependently typed programming language.

References

[1] J.-Y. Girard, "Une extension de l'interprétation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types," in *2nd Scandinavian Logic Symposium*. North-Holland, 1971, pp. 63–69.

[2] J. Reynolds, "Towards a theory of type structure," in *Programming Symposium, Paris, April 9-11, 1974*, ser. Lecture Notes in Computer Science. Springer, 1974, pp. 408–423.

[3] C. Spector, "Provably recursive functionals of analysis: a consistency proof of analysis by an extension of principles in current intuitionistic mathematics," in *Recursive Function Theory: Proceedings of Symposia in Pure Mathematics*, vol. 5. American Mathematical Society, 1962, pp. 1–27.

[4] U. Berger and P. Oliva, "Modified bar recursion and classical dependent choice," in *Logic Colloquium '01*, ser. Lecture Notes in Logic, vol. 20. Springer-Verlag, 2005, pp. 89–107.

[5] J.-L. Krivine, "Bar Recursion in Classical Realisability: Dependent Choice and Continuum Hypothesis," in *25th EACSL Annual Conference on Computer Science Logic*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016, pp. 25:1–25:11.

[6] S. Berardi, M. Bezem, and T. Coquand, "On the Computational Content of the Axiom of Choice," *Journal of Symbolic Logic*, vol. 63, no. 2, pp. 600–622, 1998.

[7] U. Berger, "The Berardi-Bezem-Coquand-functional in a domain-theoretic setting," http://www-compsci.swan.ac.uk/ csulrich/ftp/bbc.ps.gz.

[8] J. C. Reynolds, "Types, Abstraction and Parametric Polymorphism," in *IFIP Congress*, 1983, pp. 513–523.

[9] P. Wadler, "The Girard-Reynolds isomorphism (second edition)," *Theoretical Computer Science*, vol. 375, no. 1-3, pp. 201–226, 2007.

[10] U. Berger, "Program Extraction from Normalization Proofs," in *1st International Conference on Typed Lambda Calculi and Applications*. Springer, 1993, pp. 91–106.

[11] T. Altenkirch, M. Hofmann, and T. Streicher, "Reduction-Free Normalisation for a Polymorphic System," in *11th IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1996, pp. 98–106.

[12] A. Abel, "Weak beta-eta-Normalization and Normalization by Evaluation for System F," in *15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*. Springer, 2008, pp. 497–511.

[13] J.-L. Krivine, *Lambda-calculus, types and models*, ser. Ellis Horwood series in computers and their applications. Masson, 1993.

[14] W. W. Tait, "A realizability interpretation of the theory of species," in *Logic Colloquium*. Springer, 1975, pp. 240–251.

[15] J.-Y. Girard, "Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur," Ph.D. dissertation, Université Paris 7, 1972.

[16] R. Amadio and P.-L. Curien, *Domains and Lambda-Calculi*, ser. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1998, vol. 46.

[17] J. Diller and W. Nahm, "Eine Variante zur Dialectica-Interpretation der Heyting-Arithmetik endlicher Typen," *Archiv für mathematische Logik und Grundlagenforschung*, vol. 16, pp. 49–66, 1974.

APPENDIX

Proof of lemma 10. We have to prove that if $\varphi \in |\forall t\,(B\,(t) \Leftrightarrow C\,(t))|$, then:

$$[\![\lambda x.\mathtt{repl}'_A]\!]_v\,(\varphi) = [\![\mathtt{repl}'_A]\!]_{v \uplus \{x \mapsto \varphi\}} \in |A\,(B) \Leftrightarrow A\,(C)|_v$$

We write $v' = v \uplus \{x \mapsto \varphi\}$ and proceed by induction on A:

- $A\,(D) \equiv D\,(M)$: since $[\![M^\diamond]\!]_v \in \Lambda$, we have by hypothesis on φ:

$$[\![x\,M^\diamond]\!]_{v'} = \varphi\,([\![M^\diamond]\!]_v) \in |B\,(t) \Leftrightarrow C\,(t)|_{v \uplus \{t \mapsto [\![M^\diamond]\!]_v\}} = |B\,(M) \Leftrightarrow C\,(M)|_v = |A\,(B) \Leftrightarrow A\,(C)|_v$$

- $A\,(D) \equiv \Phi \not\equiv D\,(M)$: immediate since $A\,(B) \equiv A\,(C)$
- $A\,(D) \equiv A_1\,(D) \Rightarrow A_2\,(D)$: we have by induction hypothesis:

$$[\![\mathrm{p}_2\,\mathtt{repl}'_{A_1}]\!]_{v'} \in |A_1\,(C) \Rightarrow A_1\,(B)|_v$$

  therefore if $\psi \in |A\,(B)|_v$ and $\theta \in |A_1\,(C)|_v$ we get:

$$[\![\psi\,(\mathrm{p}_2\,\mathtt{repl}'_{A_1}\,\theta)]\!]_{v'} \in |A_2\,(B)|_v$$

  but the second induction hypothesis gives:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_2}]\!]_{v'} \in |A_2\,(B) \Rightarrow A_2\,(C)|_v$$

  so that we have:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_2}\,(\psi\,(\mathrm{p}_2\,\mathtt{repl}'_{A_1}\,\theta))]\!]_{v'} \in |A_2\,(C)|_v$$

  and therefore:

$$[\![\lambda yz.\mathrm{p}_1\,\mathtt{repl}'_{A_2}\,(y\,(\mathrm{p}_2\,\mathtt{repl}'_{A_1}\,z))]\!]_{v'} \in |A\,(B) \Rightarrow A\,(C)|_v$$

  similarly we have:

$$[\![\lambda yz.\mathrm{p}_2\,\mathtt{repl}'_{A_2}\,(y\,(\mathrm{p}_1\,\mathtt{repl}'_{A_1}\,z))]\!]_{v'} \in |A\,(C) \Rightarrow A\,(B)|_v$$

  and therefore $[\![\mathtt{repl}'_A]\!]_{v'} \in |A\,(B) \Leftrightarrow A\,(C)|_v$
- $A\,(D) \equiv A_1\,(D) \wedge A_2\,(D)$: we have by induction hypothesis:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_1}]\!]_{v'} \in |A_1\,(B) \Rightarrow A_1\,(C)|_v$$

  therefore if $\psi \in |A\,(B)|_v$ we get:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_1}\,(\mathrm{p}_1\,\psi)]\!]_{v'} \in |A_1\,(C)|_v$$

  but the second induction hypothesis gives:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_2}]\!]_{v'} \in |A_2\,(B) \Rightarrow A_2\,(C)|_v$$

  so that we have:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_2}\,(\mathrm{p}_2\,\psi)]\!]_{v'} \in |A_2\,(C)|_v$$

  and therefore:

$$[\![\lambda y.\,\langle \mathrm{p}_1\,\mathtt{repl}'_{A_1}\,(\mathrm{p}_1\,y)\,,\mathrm{p}_1\,\mathtt{repl}'_{A_2}\,(\mathrm{p}_2\,y)\rangle]\!]_{v'} \in |A\,(B) \Rightarrow A\,(C)|_v$$

  similarly we have:

$$[\![\lambda y.\,\langle \mathrm{p}_2\,\mathtt{repl}'_{A_1}\,(\mathrm{p}_1\,y)\,,\mathrm{p}_2\,\mathtt{repl}'_{A_2}\,(\mathrm{p}_2\,y)\rangle]\!]_{v'} \in |A\,(C) \Rightarrow A\,(B)|_v$$

  and therefore $[\![\mathtt{repl}'_A]\!]_{v'} \in |A\,(B) \Leftrightarrow A\,(C)|_v$
- $A\,(D) \equiv \forall \eta\,A_0\,(D)$: we do the case $\eta \equiv t$, the other ones being similar. The induction hypothesis implies that for any $\mathfrak{M} \in \Lambda$:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_0}]\!]_{v' \uplus \{t \mapsto \mathfrak{M}\}} \in |A_0\,(B) \Rightarrow A_0\,(C)|_{v \uplus \{t \mapsto \mathfrak{M}\}}$$

  if $\psi \in |A\,(B)|_v$ and $\mathfrak{M} \in \Lambda$ then:

$$[\![\psi\,t]\!]_{v' \uplus \{t \mapsto \mathfrak{M}\}} = \psi\,(\mathfrak{M}) \in |A_0\,(B)|_{v \uplus \{t \mapsto \mathfrak{M}\}}$$

  so that we have:

$$[\![\mathrm{p}_1\,\mathtt{repl}'_{A_0}\,(\psi\,t)]\!]_{v' \uplus \{t \mapsto \mathfrak{M}\}} \in |A_0\,(C)|_{v \uplus \{t \mapsto \mathfrak{M}\}}$$

  and therefore:

$$[\![\lambda yt.\mathrm{p}_1\,\mathtt{repl}'_{A_0}\,(y\,t)]\!]_{v'} \in |A\,(B) \Rightarrow A\,(C)|_v$$

similarly we have:

$$\left[\!\left[\lambda yt.\mathtt{p}_2\,\mathtt{repl}'_{A_0}\,(y\,t)\right]\!\right]_{v'} \in |A\,(C) \Rightarrow A\,(B)|_v$$

and therefore $[\![\mathtt{repl}'_A]\!]_{v'} \in |A\,(B) \Leftrightarrow A\,(C)|_v$

- $A\,(D) \equiv \forall X\,A_0\,(D)$ or $A\,(D) \equiv \forall b\,A_0\,(D)$: we treat only the case of X since the other one is similar. The induction hypothesis implies that for any $\mathfrak{X} \subseteq \Lambda$:

$$\left[\!\left[\mathtt{repl}'_{A_0}\right]\!\right]_{v' \uplus \{X \mapsto \mathfrak{X}\}} \in |A_0\,(B) \Leftrightarrow A_0\,(C)|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

but since $\mathtt{repl}'_{A_0}$ does not contain variable X we get:

$$\left[\!\left[\mathtt{repl}'_{A_0}\right]\!\right]_{v'} \in |A_0\,(B) \Leftrightarrow A_0\,(C)|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

so we get:

$$\left[\!\left[\mathtt{repl}'_{A_0}\right]\!\right]_{v'} \in |\forall X\,(A_0\,(B) \Leftrightarrow A_0\,(C))|_v$$

but since for any closed formulas with parameters D and D' we have:

$$|\forall X\,(D \wedge D')| = |\forall X\,D \wedge \forall X\,D'|$$
$$|\forall X\,(D \Rightarrow D')| \subseteq |\forall X\,D \Rightarrow \forall X\,D'|$$

we then obtain $[\![\mathtt{repl}'_A]\!]_{v'} \in |A\,(B) \Leftrightarrow A\,(C)|_v$ □

Proof of lemma 12.

- $\left[\!\left[\mathtt{normrc}^{(1)}\right]\!\right] \in \left|\forall\pi\,\neg\forall i\,\underline{0}\,\pi\!\downarrow^i\right|$: let $\mathfrak{p} \in \Lambda^*$ and let $\varphi \in \left|\forall i\,\underline{0}\,\mathfrak{p}\!\downarrow^i\right|$. Then $\varphi\,([\![\mathtt{z}]\!]) = \varphi\,(0) \in \left|\underline{0}\,\mathfrak{p}\!\downarrow^0\right|$. Since $\left[\!\left[(\underline{0}\,\mathfrak{p})^\diamond\right]\!\right] = [\![\mathtt{app}^*\,(\mathtt{var}\,\mathtt{z})\,\mathfrak{p}]\!]$ is in head normal form, $\left|\underline{0}\,\mathfrak{p}\!\downarrow^0\right| = |\mathit{ff}|$ and therefore $\varphi\,(0) \in |\mathit{ff}|$
- $\left[\!\left[\mathtt{normrc}^{(2)}\right]\!\right] \in |\forall t\,(t\!\downarrow \Rightarrow t\!\downarrow)|$: immediate
- $\left[\!\left[\mathtt{normrc}^{(3)}\right]\!\right] \in |\forall t\,\forall u\,\forall\pi\,((t\,[u]\,\pi)\!\downarrow \Rightarrow ((\lambda.t)\,\langle u\rangle\,\pi)\!\downarrow)|$: let $\mathfrak{M},\mathfrak{N} \in \Lambda$, $\mathfrak{p} \in \Lambda^*$, $\varphi \in |(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})\!\downarrow|$ and $\psi \in \left|\forall i\ ((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\mathfrak{p})\!\downarrow^i\right|$. We have to prove that:

$$[\![\varphi\,(\lambda i.\psi\,(\mathtt{s}\,i))]\!] \in |\mathit{ff}|$$

but since $\varphi \in |(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})\!\downarrow|$, this reduces to:

$$[\![\lambda i.\psi\,(\mathtt{s}\,i)]\!] \in \left|\forall i\ (\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})\!\downarrow^i\right|$$

Let $\mathfrak{n} \in \mathbb{N}$, we need to prove:

$$\psi\,(\mathfrak{n}+1) \in |(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})\!\downarrow^{\mathfrak{n}}|$$

But $\psi\,(\mathfrak{n}+1) \in \left|((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\mathfrak{p})\!\downarrow^{\mathfrak{n}+1}\right|$ and:

$$[\![\mathtt{app}^*\,(\mathtt{app}\,(\mathtt{abs}\,\mathfrak{M})\,\mathfrak{N})\,\mathfrak{p}]\!] \succ [\![\mathtt{app}^*\,(\mathtt{subst}\,\mathfrak{M}\,\mathtt{z}\,(\mathtt{cons}\,\mathtt{nil}\,\mathfrak{N}))\,\mathfrak{p}]\!]$$

for weak head reduction and therefore:

$$\left|((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\mathfrak{p})\!\downarrow^{\mathfrak{n}+1}\right| = |(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})\!\downarrow^{\mathfrak{n}}|$$

□

Proof of lemma 13.

- X: we have by hypothesis:

$$v'\,(x_X) \in \left|\mathcal{R}\mathit{ed}\mathcal{C}\mathit{and}\left(\overline{X}\right)\right|_v = |\mathcal{R}\mathit{ed}\mathcal{C}\mathit{and}\,(RC_X)|_v$$

therefore:

$$[\![\langle\langle \mathtt{p}_1\,(\mathtt{p}_1\,x_X)\,,\mathtt{p}_2\,(\mathtt{p}_1\,x_X)\rangle\,,\mathtt{p}_2\,x_X\rangle]\!]_{v'} = [\![x_X]\!]_{v'} \in |\mathcal{R}\mathit{ed}\mathcal{C}\mathit{and}\,(RC_X)|_v$$

- $T \to U$:
  - $\left[\!\left[\mathtt{isrc}^{(1)}_{T\to U}\right]\!\right]_{v'} \in |\forall\pi\,RC_{T\to U}\,(\underline{0}\,\pi)|_v$: let $\mathfrak{p} \in \Lambda^*$, $\mathfrak{M} \in \Lambda$ and $\varphi \in |RC_T\,(\mathfrak{M})|_v$. The induction hypothesis gives:

$$\left[\!\left[\mathtt{isrc}^{(1)}_U\right]\!\right]_{v'} \in |\forall\pi\,RC_U\,(\underline{0}\,\pi)|$$

therefore:

$$\left[\!\left[\mathtt{isrc}^{(1)}_U\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{M})\right]\!\right]_{v'} \in |RC_U\,(\underline{0}\,\langle\mathfrak{p},\mathfrak{M}\rangle)|$$

  - $\left[\!\left[\mathtt{isrc}^{(2)}_{T\to U}\right]\!\right]_{v'} \in |\forall t\,(RC_{T\to U}\,(t) \Rightarrow t\!\downarrow)|_v$: let $\mathfrak{M} \in \Lambda$ and $\varphi \in |RC_{T\to U}\,(\mathfrak{M})|_v$. The induction hypothesis implies:

$$\left[\!\left[\mathtt{isrc}^{(1)}_T\,\mathtt{nil}\right]\!\right]_{v'} \in |RC_T\,(\underline{0}\,\langle\rangle)|_v = |RC_T\,(\underline{0})|_v$$

so since $[\![\varphi\,(\mathtt{var}\,\mathtt{z})]\!] \in |RC_T\,(\underline{0}) \Rightarrow RC_U\,(\mathfrak{M}\,\langle\underline{0}\rangle)|_v$ we get:

$$\left[\!\left[\varphi\,(\mathtt{var}\,\mathtt{z})\left(\mathtt{isrc}^{(1)}_T\,\mathtt{nil}\right)\right]\!\right]_{v'} \in |RC_U\,(\mathfrak{M}\,\langle\underline{0}\rangle)|_v$$

The second induction hypothesis implies:

$$\left[\left[\mathtt{isrc}_U^{(2)}\,(\mathtt{app}\,\mathfrak{M}\,(\mathtt{var}\,\mathtt{z}))\right]\right]_{v'} \in |RC_U\,(\mathfrak{M}\,\langle\underline{0}\rangle) \Rightarrow \mathfrak{M}\,\langle\underline{0}\rangle\downarrow|_v$$

and therefore:

$$\left[\left[\mathtt{isrc}_U^{(2)}\,(\mathtt{app}\,\mathfrak{M}\,(\mathtt{var}\,\mathtt{z}))\,\left(\varphi\,(\mathtt{var}\,\mathtt{z})\,\left(\mathtt{isrc}_T^{(1)}\,\mathtt{nil}\right)\right)\right]\right]_{v'} \in |\mathfrak{M}\,\langle\underline{0}\rangle\downarrow|_v$$

To conclude, we prove that $|\mathfrak{M}\,\langle\underline{0}\rangle\downarrow|_v \subseteq |\mathfrak{M}\downarrow|_v$. This follows from $\left|\forall i\,\mathfrak{M}\downarrow^i\right|_v \subseteq \left|\forall i\,\mathfrak{M}\,\langle\underline{0}\rangle\downarrow^i\right|_v$, which follows from $|\mathfrak{M}\downarrow^{\mathfrak{n}}|_v \subseteq |\mathfrak{M}\,\langle\underline{0}\rangle\downarrow^{\mathfrak{n}}|_v$. This last inclusion comes from the fact that if $\mathfrak{M}$ does not reach a normal form in $\mathfrak{n}$ steps, then $\mathfrak{M}\,0$ does not reach a normal form in $\mathfrak{n}$ steps either.

– $\left[\left[\mathtt{isrc}_{T\to U}^{(3)}\right]\right]_{v'} \in |\forall t\,\forall u\,\forall\pi\,(RC_{T\to U}\,(t\,[u]\,\pi) \Rightarrow RC_{T\to U}\,(\lambda.t\,u\,\pi))|_v$: let $\mathfrak{M} \in \Lambda$, $\mathfrak{N} \in \Lambda$, $\mathfrak{p} \in \Lambda^*$, $\varphi \in |RC_{T\to U}\,(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})|_v$, $\mathfrak{P} \in \Lambda$ and $\psi \in |RC_T\,(\mathfrak{P})|_v$. The induction hypothesis implies:

$$\left[\left[\mathtt{isrc}_U^{(3)}\,\mathfrak{M}\,\mathfrak{N}\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{P})\right]\right]_{v'} \in |RC_U\,(\mathfrak{M}\,[\mathfrak{N}]\,\langle\mathfrak{p},\mathfrak{P}\rangle) \Rightarrow RC_U\,((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\langle\mathfrak{p},\mathfrak{P}\rangle)|_v$$

We also have:

$$[\![\varphi\,\mathfrak{P}\,\psi]\!] \in |RC_U\,(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p}\,\langle\mathfrak{P}\rangle)|_v$$

so since:

$$[\![\mathtt{app}\,(\mathtt{app}^*\,(\mathtt{subst}\,\mathfrak{M}\,\mathtt{z}\,(\mathtt{cons}\,\mathtt{nil}\,\mathfrak{N}))\,\mathfrak{p})\,\mathfrak{P}]\!] = [\![\mathtt{app}^*\,(\mathtt{subst}\,\mathfrak{M}\,\mathtt{z}\,(\mathtt{cons}\,\mathtt{nil}\,\mathfrak{N}))\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{P})]\!]$$

we have:

$$|RC_U\,(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p}\,\langle\mathfrak{P}\rangle)|_v = |RC_U\,(\mathfrak{M}\,[\mathfrak{N}]\,\langle\mathfrak{p},\mathfrak{P}\rangle)|_v$$

and therefore:

$$\left[\left[\mathtt{isrc}_U^{(3)}\,\mathfrak{M}\,\mathfrak{N}\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{P})\,(\varphi\,\mathfrak{P}\,\psi)\right]\right]_{v'} \in |RC_U\,((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\langle\mathfrak{p},\mathfrak{P}\rangle)|_v$$

and finally since:

$$[\![\mathtt{app}^*\,(\mathtt{app}\,(\mathtt{abs}\,\mathfrak{M})\,\mathfrak{N})\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{P})]\!] = [\![\mathtt{app}\,(\mathtt{app}^*\,(\mathtt{app}\,(\mathtt{abs}\,\mathfrak{M})\,\mathfrak{N})\,\mathfrak{p})\,\mathfrak{P}]\!]$$

we obtain:

$$\left[\left[\mathtt{isrc}_U^{(3)}\,\mathfrak{M}\,\mathfrak{N}\,(\mathtt{cons}\,\mathfrak{p}\,\mathfrak{P})\,(\varphi\,\mathfrak{P}\,\psi)\right]\right]_{v'} \in |RC_U\,((\lambda.\mathfrak{M})\,\langle\mathfrak{N}\rangle\,\mathfrak{p}\,\langle\mathfrak{P}\rangle)|_v$$

- $\forall X\,T$:
  – $\left[\left[\mathtt{isrc}_{\forall X\,T}^{(1)}\right]\right]_{v'} \in |\forall\pi\,RC_{\forall X\,T}\,(\underline{0}\,\pi)|_v$: let $\mathfrak{p} \in \Lambda^*$, $\mathfrak{X} \subseteq \Lambda$ and $\varphi \in \left|\mathcal{R}ed\mathcal{C}and\,(\overline{X})\right|_{v\uplus\{X\mapsto\mathfrak{X}\}}$. Then by induction hypothesis:

$$\left[\left[\mathtt{isrc}_T^{(1)}\,\mathfrak{p}\right]\right]_{v'\uplus\{x_X\mapsto\varphi\}} \in |RC_T\,(\underline{0}\,\mathfrak{p})|_{v\uplus\{X\mapsto\mathfrak{X}\}}$$

therefore:

$$\left[\left[\lambda x_X.\mathtt{isrc}_T^{(1)}\,\mathfrak{p}\right]\right]_{v'} \in \left|\mathcal{R}ed\mathcal{C}and\,(\overline{X}) \Rightarrow RC_T\,(\underline{0}\,\mathfrak{p})\right|_{v\uplus\{X\mapsto\mathfrak{X}\}}$$

and finally:

$$\left[\left[\lambda x_X.\mathtt{isrc}_T^{(1)}\,\mathfrak{p}\right]\right]_{v'} \in |RC_{\forall X\,T}\,(\underline{0}\,\mathfrak{p})|_v$$

  – $\left[\left[\mathtt{isrc}_{\forall X\,T}^{(2)}\right]\right]_{v'} \in |\forall t\,(RC_{\forall X\,T}\,(t) \Rightarrow t\downarrow)|_v$: let $\mathfrak{M} \in \Lambda$ and $\varphi \in |RC_{\forall X\,T}\,(t)|_{v\uplus\{t\mapsto\mathfrak{M}\}}$. The induction hypothesis implies that for any $\mathfrak{X} \subseteq \Lambda$:

$$\left[\left[\lambda x_X.\mathtt{isrc}_T^{(2)}\right]\right]_{v'} \in \left|\mathcal{R}ed\mathcal{C}and\,(\overline{X}) \Rightarrow \forall t\,(RC_T\,(t) \Rightarrow t\downarrow)\right|_{v\uplus\{X\mapsto\mathfrak{X}\}}$$

therefore:

$$\left[\left[\lambda x_X.\mathtt{isrc}_T^{(2)}\right]\right]_{v'} \in \left|\forall X\,\left(\mathcal{R}ed\mathcal{C}and\,(\overline{X}) \Rightarrow \forall t\,(RC_T\,(t) \Rightarrow t\downarrow)\right)\right|_v$$

and so by lemma 11:

$$\left[\left[\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}ed\mathcal{C}and(\overline{X})\Rightarrow\forall t(RC_T(t)\Rightarrow t\downarrow),\Downarrow}\,\left(\lambda x_X.\mathtt{isrc}_T^{(2)}\right)\right]\right]_{v'} \in \left|\mathcal{R}ed\mathcal{C}and\,(\overline{\Downarrow}) \Rightarrow \forall t\,\left((\overline{X}\mapsto RC_T\,(t))\,(\Downarrow) \Rightarrow t\downarrow\right)\right|_v$$

and then by lemma 12:

$$\left[\left[\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}ed\mathcal{C}and(\overline{X})\Rightarrow\forall t(RC_T(t)\Rightarrow t\downarrow),\Downarrow}\,\left(\lambda x_X.\mathtt{isrc}_T^{(2)}\right)\,\mathtt{normrc}\,t\right]\right]_{v\uplus\{t\mapsto\mathfrak{M}\}} \in \left|(\overline{X}\mapsto RC_T\,(t))\,(\Downarrow) \Rightarrow t\downarrow\right|_{v\uplus\{t\mapsto\mathfrak{M}\}}$$

On the other hand, since:

$$\varphi \in \left|\forall X\,\left(\mathcal{R}ed\mathcal{C}and\,(\overline{X}) \Rightarrow RC_T\,(t)\right)\right|_{v\uplus\{t\mapsto\mathfrak{M}\}}$$

and since by lemma 11, for any $\mathfrak{M} \in \lambda$:

$$\left[\!\left[\mathtt{elim}_{\overline{X} \mapsto \mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T(t), \Downarrow}\right]\!\right]_{v \uplus \{t \mapsto \mathfrak{M}\}} \in \left|\forall X \left(\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T(t)\right) \Rightarrow \mathcal{R}ed\mathcal{C}and\left(\overline{\Downarrow}\right) \Rightarrow \left(\overline{X} \mapsto RC_T(t)\right)(\Downarrow)\right|_{v \uplus \{t \mapsto \mathfrak{M}\}}$$

we have:

$$\left[\!\left[\mathtt{elim}_{\overline{X} \mapsto \mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T(t), \Downarrow} \varphi\, \mathtt{normrc}\right]\!\right]_{v \uplus \{t \mapsto \mathfrak{M}\}} \in \left|\left(\overline{X} \mapsto RC_T(t)\right)(\Downarrow)\right|_{v \uplus \{t \mapsto \mathfrak{M}\}}$$

and therefore we get:

$$\left[\!\left[\mathtt{elim}_{\overline{X} \mapsto \mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow \forall t (RC_T(t) \Rightarrow \not\Downarrow), \Downarrow} \left(\lambda x_X.\mathtt{isrc}_T^{(2)}\right) \mathtt{normrc}\, t \left(\mathtt{elim}_{\overline{X} \mapsto \mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T(t), \Downarrow} \varphi\, \mathtt{normrc}\right)\right]\!\right]_{v \uplus \{t \mapsto \mathfrak{M}\}} \in |t\!\downarrow|_{v \uplus \{t \mapsto \mathfrak{M}\}} = |\mathfrak{M}\!\downarrow|$$

– $\left[\!\left[\mathtt{isrc}^{(3)}_{\forall X\, T}\right]\!\right]_{v'} \in |\forall t \forall u \forall \pi \left(RC_{\forall X\, T}(t\,[u]\,\pi) \Rightarrow RC_{\forall X\, T}(\lambda.t\, u\, \pi)\right)|_v$: let $\mathfrak{M} \in \Lambda$, $\mathfrak{N} \in \Lambda$, $\mathfrak{p} \in \Lambda^*$, $\varphi \in |RC_{\forall X\, T}(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})|_v$, $\mathfrak{X} \subseteq \Lambda$ and $\psi \in \left|\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right)\right|_{v \uplus \{X \mapsto \mathfrak{X}\}}$. The induction hypothesis implies:

$$\left[\!\left[\mathtt{isrc}^{(3)}_T\, \mathfrak{M}\, \mathfrak{N}\, \mathfrak{p}\right]\!\right]_{v \uplus \{x_X \mapsto \psi\}} \in |RC_T(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p}) \Rightarrow RC_T(\lambda.\mathfrak{M}\,\mathfrak{N}\,\mathfrak{p})|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

but we have also:

$$[\![\varphi\, \psi]\!] \in |RC_T(\mathfrak{M}\,[\mathfrak{N}]\,\mathfrak{p})|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

therefore:

$$\left[\!\left[\mathtt{isrc}^{(3)}_T\, \mathfrak{M}\, \mathfrak{N}\, \mathfrak{p}\, (\varphi\, \psi)\right]\!\right]_{v \uplus \{x_X \mapsto \psi\}} \in |RC_T(\lambda.\mathfrak{M}\,\mathfrak{N}\,\mathfrak{p})|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

and therefore:

$$\left[\!\left[\lambda x_X.\mathtt{isrc}^{(3)}_T\, \mathfrak{M}\, \mathfrak{N}\, \mathfrak{p}\, (\varphi\, x_X)\right]\!\right]_{v'} \in \left|\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T(\lambda.\mathfrak{M}\,\mathfrak{N}\,\mathfrak{p})\right|_{v \uplus \{X \mapsto \mathfrak{X}\}}$$

and since this holds for any $\mathfrak{X} \subseteq \Lambda$ we obtain:

$$\left[\!\left[\lambda x_X.\mathtt{isrc}^{(3)}_T\, \mathfrak{M}\, \mathfrak{N}\, \mathfrak{p}\, (\varphi\, x_X)\right]\!\right]_{v'} \in |RC_{\forall X\, T}(\lambda.\mathfrak{M}\,\mathfrak{N}\,\mathfrak{p})|_v \qquad \square$$

Proof of theorem 2.

- $\Gamma \vdash \underline{m} : U$: the hypothesis gives:

$$v(y_U) \in |RC_U(t_U)|_v$$

but since $\left[\!\left[\mathtt{subst}\,(\mathtt{var}\,(\mathtt{s}^m\,\mathtt{z}))\,\mathtt{z}\,\vec{t_\Gamma}\right]\!\right]_v = [\![t_U{}^\diamond]\!]_v$ we have:

$$\left|RC_U\left(\underline{m}\left[\vec{t_\Gamma}\right]\right)\right|_v = |RC_U(t_U)|_v$$

- $\Gamma \vdash \lambda.M : U \to T$: let $\mathfrak{M} \in \Lambda$ and $\varphi \in |RC_U(\mathfrak{M})|_v$. If we write $\mathfrak{N} = \left[\!\left[\mathtt{subst}\, M^\diamond\, (\mathtt{s}\,\mathtt{z})\left(\mathtt{shift}^*\, \vec{t_\Gamma}\right)\right]\!\right]_v \in \Lambda$, then lemma 13 implies:

$$\left[\!\left[\mathtt{isrc}^{(3)}_T\, \mathfrak{N}\, \mathfrak{M}\, \mathtt{nil}\right]\!\right]_v \in |RC_T(\mathfrak{N}\,[\mathfrak{M}]\,\langle\rangle) \Rightarrow RC_T((\lambda.\mathfrak{N})\,\mathfrak{M}\,\langle\rangle)|_v$$

On the other hand, the induction hypothesis implies:

$$\left[\!\left[\mathtt{adeq}_{\Gamma, U \vdash M:T}\right]\!\right]_{v \uplus \{t_U \mapsto \mathfrak{M}; x_U \mapsto \varphi\}} \in \left|RC_T\left(M\left[t_{\Gamma,U}^{\vec{}}\right]\right)\right|_{v \uplus \{t_U \mapsto \mathfrak{M}\}}$$

but since we have by a version of lemma 1 in system ΛT_{bbc}:

$$\begin{aligned}\left[\!\left[(\mathfrak{N}\,[\mathfrak{M}]\,\langle\rangle)^\diamond\right]\!\right]_v &= \left[\!\left[\mathtt{subst}\left(\mathtt{subst}\, M^\diamond\, (\mathtt{s}\,\mathtt{z})\left(\mathtt{shift}^*\, \vec{t_\Gamma}\right)\right)\mathtt{z}\,(\mathtt{cons}\,\mathtt{nil}\,\mathfrak{M})\right]\!\right]_v \\ &= \left[\!\left[\mathtt{subst}\, M^\diamond\, \mathtt{z}\, t_{\Gamma,U}^{\vec{}}\right]\!\right]_{v \uplus \{t_U \mapsto \mathfrak{M}\}} \\ &= \left[\!\left[\left(M\left[t_{\Gamma,U}^{\vec{}}\right]\right)^\diamond\right]\!\right]_{v \uplus \{t_U \mapsto \mathfrak{M}\}}\end{aligned}$$

we also have:

$$|RC_T(\mathfrak{N}\,[\mathfrak{M}]\,\langle\rangle)|_v = \left|RC_T\left(M\left[t_{\Gamma,U}^{\vec{}}\right]\right)\right|_{v \uplus \{t_U \mapsto \mathfrak{M}\}}$$

and therefore:

$$\left[\!\left[\mathtt{isrc}^{(3)}_T\left(\mathtt{subst}\, M^\diamond\, (\mathtt{s}\,\mathtt{z})\left(\mathtt{shift}^*\, \vec{t_\Gamma}\right)\right) t_U\, \mathtt{nil}\, \mathtt{adeq}_{\Gamma, U \vdash M:T}\right]\!\right]_{v \uplus \{t_U \mapsto \mathfrak{M}; x_U \mapsto \varphi\}} \in |RC_T((\lambda.\mathfrak{N})\,\mathfrak{M})|_v$$

- $\Gamma \vdash M\ N : T$: the first induction hypothesis implies:

$$\left[\!\left[\mathtt{adeq}_{\Gamma\vdash M:U\to T}\left(\mathtt{subst}\ N^{\diamond}\ \mathtt{z}\ \vec{t_\Gamma}\right)\right]\!\right]_v \in \left|RC_U\left(N\left[\vec{t_\Gamma}\right]\right) \Rightarrow RC_T\left(M\left[\vec{t_\Gamma}\right]\left\langle N\left[\vec{t_\Gamma}\right]\right\rangle\right)\right|_v$$

and the second induction hypothesis gives:

$$[\![\mathtt{adeq}_{\Gamma\vdash N:U}]\!]_v \in \left|RC_U\left(N\left[\vec{t_\Gamma}\right]\right)\right|_v$$

so since:

$$\left[\!\left[\mathtt{app}\left(\mathtt{subst}\ M^{\diamond}\ \mathtt{z}\ \vec{t_\Gamma}\right)\left(\mathtt{subst}\ N^{\diamond}\ \mathtt{z}\ \vec{t_\Gamma}\right)\right]\!\right]_v = \left[\!\left[\mathtt{subst}\left(\mathtt{app}\ M^{\diamond}\ M^{\diamond}\right)\ \mathtt{z}\ \vec{t_\Gamma}\right]\!\right]_v$$

we obtain:

$$\left[\!\left[\mathtt{adeq}_{\Gamma\vdash M:U\to T}\left(\mathtt{subst}\ N^{\diamond}\ \mathtt{z}\ \vec{t_\Gamma}\right)\mathtt{adeq}_{\Gamma\vdash N:U}\right]\!\right]_v \in \left|RC_T\left(\left(M\left\langle N\right\rangle\right)\left[\vec{t_\Gamma}\right]\right)\right|_v$$

- $\Gamma \vdash M : \forall X\ T$: let $\mathfrak{X} \subseteq \Lambda$ and $\varphi \in \left|\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right)\right|_{v\uplus\{X\mapsto\mathfrak{X}\}}$. Then the induction hypothesis gives immediately:

$$[\![\mathtt{adeq}_{\Gamma\vdash M:T}]\!]_{v\uplus\{X\mapsto\mathfrak{X};x_X\mapsto\varphi\}} \in \left|RC_T\left(M\left[\vec{t_\Gamma}\right]\right)\right|_{v\uplus\{X\mapsto\mathfrak{X};x_X\mapsto\varphi\}}$$

- $\Gamma \vdash M : T\{U/X\}$: the induction hypothesis gives:

$$[\![\mathtt{adeq}_{\Gamma\vdash M:\forall X\ T}]\!]_v \in \left|\forall X\left(\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right) \Rightarrow RC_T\left(M\left[\vec{t_\Gamma}\right]\right)\right)\right|_v$$

therefore lemma 11 implies:

$$\left[\!\left[\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right)\Rightarrow RC_T\left(M\left[\vec{t_\Gamma}\right]\right),RC_U}\mathtt{adeq}_{\Gamma\vdash M:\forall X\ T}\right]\!\right]_v \in \left|\mathcal{R}ed\mathcal{C}and\left(RC_U\right) \Rightarrow \left(\overline{X} \mapsto RC_T\left(M\left[\vec{t_\Gamma}\right]\right)\right)\left(RC_U\right)\right|_v$$

and since by lemma 13 we have:

$$[\![\mathtt{isrc}_U]\!]_v \in \left|\mathcal{R}ed\mathcal{C}and\left(RC_U\right)\right|_v$$

and moreover for any M:

$$\left(\overline{X} \mapsto RC_T\left(M\right)\right)\left(RC_U\right) \equiv RC_{T\{U/X\}}\left(M\right)$$

we obtain:

$$\left[\!\left[\mathtt{elim}_{\overline{X}\mapsto\mathcal{R}ed\mathcal{C}and\left(\overline{X}\right)\Rightarrow RC_T\left(M\left[\vec{t_\Gamma}\right]\right),RC_U}\mathtt{adeq}_{\Gamma\vdash M:\forall X\ T}\ \mathtt{isrc}_U\right]\!\right]_v \in \left|RC_{T\{U/X\}}\left(M\left[\vec{t_\Gamma}\right]\right)\right|_v \qquad \square$$