Completeness, minimal logic and programs extraction

Christophe Raffalli

LAMA - Équipe de Logique, Université de Savoie
73376 Le Bourget du Lac
email: Christophe.Raffalli@univ-savoie.fr

Abstract

We formalize, in the AF_2 type system, a completeness and correctness theorem for a modified Kripke semantics for minimal logic. We show that any program extracted from a proof of the completeness theorem translates an higher-order encoding of a λ-term into a Debruijn encoding of another λ-term with the same type. We also show that any program extracted from the correctness theorem performs the reverse translation. Moreover in both cases, there is one proof which corresponds to a program that does not change the underlying λ-term.

Key words: λ-calculus. Types. AF_2 type system. Intuitionnistic Logic. Program Extraction.

1 Introduction

The Curry-Howard isomorphism [2] establishes a correspondence between proofs and programs. We can extract a correct program from a proof of its specification. But to what specification corresponds a mathematical theorem like the completeness theorem ? More precisely, is there a property characterizing any program extracted from a proof of the completeness theorem. We will answer this question for the case of minimal logic [1] using a variation on Kripke semantics [3].

[1] I shall thank Jean-Louis Krivine for his comments and the fruitful discussions we had.

We formalize the syntax and the semantics of minimal logic in the AF_2 type system [4] (This system is the second order intuitionnistic logic with program extraction). We prove (theorem 9) that a λ-term extracted from a proof of the provability of a formula F in minimal logic is a *Debruijn representation* of a λ-term of type F (in simply-typed λ-calculus). We mean by a Debruijn representation an encoding of λ-terms in λ-calculus where a bound variable is encoded by the number of abstractions separating it from its binder.

We establish a similar result (theorem 26) for a proof that a formula F is true in every models, but instead of a Debruijn representation, we get a *higher-order representation*, where bound variables are encoded using λ-abstractions. The semantics we use is a variation on the Kripke semantics [3]. The main difference is the use of a binary function instead of an ordering. This function can be seen as the concatenation of two paths in a Kripke tree. Using functions allows us to use equationnal axioms which enjoy a trivial algorithmic contents.

From this two results, we deduce (theorem 27) that any term extracted from a proof of the completeness theorem 23 translates an higher-order encoding of a simply typed λ-term to the Debruijn encoding of a term of the same type. We also get the reverse translation from any proof of the correctness theorem 14.

An important point in this work is the precise choice of the definition to simplify as much as possible the term and get only the essential algorithmic meaning of the theorem. Indeed, the natural definition of provable and true would have had more information (all quantifications would have been restricted to formulas or contexts) inducing some noise in the extracted program.

This simplification of the definitions results in a more general completeness and correctness theorems than usual. Indeed, we do not have to know that formulas and contexts are well founded. However, we need some equational axioms that could be proven if we added the hypothesis that contexts and formulas were well founded.

Our result is similar to Krivine's result for first-order classical logic [6], with one main difference: we find a proof of the completeness theorem whose extracted term translates an higher-order encoding of any λ-term t to the Debruijn encoding of the same term (see theorem 25). We think that the same result is impossible for the classical logic or the intuitionnistic logic with the disjunction or the existential quantifier (for semantics similar to the usual Kripke semantics). A proof of these conjectures would give a concrete argument to say that the semantics of minimal logic is much simpler and nearer to the syntax than the semantics of classical logic or the Kripke semantics for the full intuitionnistic logic. This let us hope that programs extraction could be a tool to compare the strength of similar theorems which are not formally

related.

One should also note that the completeness and correctness theorems have been machine checked using the author's implementation of the AF_2 type system. However, the proof about the extracted terms have not been machine checked (this would require a formalization of the AF_2 realizability yet to be done).

2 Preliminaries

We use Krivine's notation for the λ-calculus (except for the application were we use the standard LISP notation) and his notation and results for the AF_2 type system [4].

To avoid confusion, we use $\Gamma \vdash_S t : F$ for sequent in simply-typed λ-calculus and $\Gamma \vdash_{AF_2} t : F$ for AF_2 sequent. Moreover, we use the following abbreviations and notations:

- $A \to B \to C$ for $A \to (B \to C)$
- $\forall x A \to B$ for $(\forall x A) \to B$
- $\forall x, y{:}A\ B$ for $\forall x(A(x) \to \forall y(A(y) \to B))$.
- Λ is the set of all λ-terms.
- $\mathcal{P}_\beta(\Lambda)$ is the set of subsets of Λ closed under β-equivalence

We recall the following definitions and lemma from [4]:

Definition 1 (Interpretation) *An* interpretation σ*: is given by three mappings:*

- *A mapping from first order variables to λ-terms denoted $x \mapsto |x|^\sigma$*
- *A mapping from function symbols of arity n to n-ary functions from Λ^n to Λ denoted $f \mapsto |f|^\sigma$*
- *A mapping from predicate variables of arity n to n-ary functions from Λ^n to $\mathcal{P}(\Lambda)_\beta$ denoted $X \mapsto |X|^\sigma$*

As usual, we write $\sigma[x \Leftarrow t]$ or $\sigma[X \Leftarrow \Phi]$ when we change the value of the mapping σ for x or X only. This mapping is extended to any term and formula by:

- $|f(t_1, \ldots, t_n)|^\sigma = |f|^\sigma(|t_1|^\sigma, \ldots, |t_n|^\sigma)$
- $|X(t_1, \ldots, t_n)|^\sigma = |X|^\sigma(|t_1|^\sigma, \ldots, |t_n|^\sigma)$
- $|A \to B|^\sigma = \{t \in \Lambda; \forall u \in |A|^\sigma, (t\ u) \in |B|^\sigma\}$
- $|\forall x\ A|^\sigma = \bigcap_{t \in \Lambda} |A|^{\sigma[x \Leftarrow t]}$
- $|\forall X\ A|^\sigma = \bigcap_{\Phi \in \Lambda^n \to \mathcal{P}_\beta(\Lambda)} |A|^{\sigma[X \Leftarrow \Phi]}$

Lemma 2 (Adequation lemma) *For any formula A and any interpretation σ, we have*

$$\vdash_{AF_2} t : F \text{ implies } t \in |F|^\sigma$$

Definition 3 (Data types) *A predicate D of arity one is a* data-type[2] *for an interpretation σ if and only if for any first order term v and any λ-term t, $t \in |D(v)|^\sigma$ implies $t \sim_\beta |v|^\sigma$.*

3 The formalization of minimal logic

Definition 4 *Let us choose an infinite countable set of variables $\mathcal{V}$, one unary function symbol $x \mapsto \overline{x}$ and one binary function symbol $f \Rightarrow g$. We define the set of formulas as the smallest set such that*[3] *:*

$$\forall x{:}\mathcal{V}\ \mathcal{F}(\overline{x}) \quad \textit{and} \quad \forall f,g{:}\mathcal{F}\ \mathcal{F}(f \Rightarrow g)$$

This is formally defined by the following second-order formula:

$$\mathcal{F}(f) = \forall X\ (\forall x{:}\mathcal{V}\ X(\overline{x}) \to \forall g,h\ (X(g) \to X(h) \to X(g \Rightarrow h)) \to X(f))$$

Definition 5 *We define contexts as ordered lists. We use a symbol $\emptyset$ for the empty list and a binary function symbol* cons *written $a, y \mapsto ay$ to construct lists. The predicate defining lists of elements of D is:*

$$\mathbb{L}_D(l) = \forall X\ (X(\emptyset) \to \forall a{:}D\ \forall y{:}X\ X(ay) \to X(l))$$

Theorem 6 *The predicates $\mathcal{F}$ and $\mathbb{L}_\mathcal{F}$ are data-types (definition 3) for an interpretation σ if the following holds:*

- *The predicate $\mathcal{V}$ defining variables is a data-type for σ*
- $|\overline{x}|^\sigma \sim_\beta \lambda a \lambda b (a\, |x|^\sigma)$
- $|f \Rightarrow g|^\sigma \sim_\beta \lambda a \lambda b (b\, |f|^\sigma\, |g|^\sigma)$
- $|\emptyset|^\sigma \sim_\beta \lambda a \lambda b\, a$
- $|xl|^\sigma \sim_\beta \lambda a \lambda b (b\, |x|^\sigma\, |l|^\sigma)$

Proof: The proofs for lists and trees can be found in [4] and the proof for formulas is identical to the proof for trees. ■

[2] Krivine also states that $|D(v)|^\sigma$ is non empty if and only if t is intentionnally of type D (for integer this means that if $|\mathbb{N}(v)|^\sigma$ is non empty then v really represents an integer). We do not use this, this is why we give a simplified definition.

[3] We cannot identify the formula x and the variable x if we want the proposition 6 to hold

Definition 7 *We define the provability as the smallest binary relation* $\Vdash$ *such that:*

$$\begin{array}{ll}
\forall C,f\ (fC \Vdash f) & \textit{(axiom)} \\
\forall C,f,g\ (C \Vdash f \to gC \Vdash f) & \textit{(weakening)} \\
\forall C,f,g\ (fC \Vdash g \to C \Vdash f \Rightarrow g) & \textit{(implication introduction)} \\
\forall C,f,g\ (C \Vdash f \Rightarrow g \to C \Vdash f \to C \Vdash g) & \textit{(implication elimination)}
\end{array}$$

Formally this predicate is defined by the following second-order predicate:

$$C \Vdash f = \forall X \begin{pmatrix} \forall f,C\ X(fC,f) \to \forall f,g,C\ (X(C,f) \to X(gC,f)) \to \\ \forall f,g,C\ (X(fC,g) \to X(C,f \Rightarrow g)) \to \\ \forall f,g,C\ (X(C,f \Rightarrow g) \to X(C,f) \to X(C,g)) \to \\ \quad X(C,f) \end{pmatrix}$$

One should note that with this definition, $C \Vdash f$ does not imply that C is a list of formulas or f is a formula. We could add more conditions in the definition, but the λ-term extracted from a proof of $C \Vdash f$ will then be more complex.

Definition 8 *We say that a* λ*-term* t *is an internal Debruijn representation of a closed term* u *if* $t = \lambda z \lambda s \lambda l \lambda a\, t'$ *and* $||t'||_\emptyset = u$ *where* $||t'||_c$ *is the partial function taking as arguments a term and a list of variables and defined by*

- $||z||_{xc} = x$.
- $||(s\, t_1)||_{xc} = ||t_1||_c$.
- $||(l\, t_1)||_c = \lambda x ||t_1||_{xc}$ *where* $x \notin c$.
- $||(a\, t_1\, t_2)||_c = (||t_1||_c\, ||t_2||_c)$.

Formally, the definition of $||t'||_c$ *depends on the choice of the variables* z, s, l, a *but we omit these parameters in the notation.*

You may notice that the internal Debruijn representations of a term u are not unique because of the freedom in the position of the weakening denoted by the variable s (usually, weakenings only occur at the variable level). For instance, $\lambda z \lambda s \lambda l \lambda a\, (l\, (l\, (s\, (a\, z\, z))))$ and $\lambda z \lambda s \lambda l \lambda a\, (l\, (l\, (a\, (s\, z)(s, z))))$ are two internal Debruijn representations for the term $\lambda x \lambda y (x\, x)$.

Theorem 9 *For any term* t *and any formula* f*, if* $\vdash_{AF_2} t : (\emptyset \Vdash f)$ *then* t *is* β*-equivalent to an internal Debruijn representation of a closed* λ*-term* u *such that* $\vdash_S u : f$.

This theorem means that a term t extracted form a proof, in second-order logic, that a formula f is provable in minimal logic is an internal Debruijn representation of a term u of type f in simply-typed λ-calculus.

Proof: To prove this theorem, we use the adequation lemma, which implies that for any interpretation σ we have $t \in |\emptyset \Vdash f|^\sigma$.

We choose an interpretation σ such that for any first-order terms ϕ and ϕ' designing formulas or lists of formulas, we have $|\phi|^\sigma \sim_\beta |\phi'|^\sigma$ implies $\phi = \phi'$ (i). Such an interpretation exists, we just need to choose σ such that the type of formulas and lists of formulas are data-types using proposition 6.

We choose four λ-variables z, s, a, l not free in t and we define, for any λ-terms t, t', $\Phi(t, t')$ the smallest sets closed under β-equivalence such that:

- $z \in \Phi(|fC|^\sigma, |f|^\sigma)$
- $(s\, t_1) \in \Phi(|gC|^\sigma, |f|^\sigma)$ if $t_1 \in \Phi(|\,C|^\sigma, |f|^\sigma)$
- $(l\, t_1) \in \Phi(|\,C|^\sigma, |f \Rightarrow g|^\sigma)$ if $t_1 \in \Phi(|fC|^\sigma, |g|^\sigma)$
- $(a\, t_1\, t_2) \in \Phi(|\,C|^\sigma, |g|^\sigma)$ if $t_1 \in \Phi(|\,C|^\sigma, |f|^\sigma)$ and $t_2 \in \Phi(|\,C|^\sigma, |f \Rightarrow g|^\sigma)$

We first show, by induction on the definition of Φ, that $t_1 \in \Phi(|C|^\sigma, |f\,|^\sigma)$, t_1 normal, $l = x_1 \ldots x_n$ with the x_i distinct and $C = f_1 \ldots f_n$ implies $x_1 : f_1, \ldots, x_n : f_n \vdash_S ||t_1||_l : f$ (ii). Each case in the induction is immediate using the property (i) and the definition of Φ and $||t_1||_l$.

Then we define the interpretation $\sigma' = \sigma[X/\Phi]$ and we prove the following properties:

- $z \in |\forall f, C\ X(fC,f)|^{\sigma'}$
- $s \in |\forall f, g, C\ (X(C,f) \to X(gC,f))|^{\sigma'}$
- $l \in |\forall f, g, C\ (X(fC,g) \to X(C,f \Rightarrow g))|^{\sigma'}$
- $a \in |\forall f, g\ \forall C\ (X(C,f \Rightarrow g) \to X(C,f) \to X(C,g))|^{\sigma'}$

This is immediate: when we apply the definition of the interpretation, each condition corresponds to one of the condition in the definition of Φ.

Then, we have $t \in |\emptyset \Vdash f|^\sigma$ implies $(t\, z\, s\, l\, a) \in |X(C,f)|^{\sigma'} = \Phi(|\emptyset|^\sigma, |f\,|^\sigma)$. Thus, using (ii) we find $(t\, z\, s\, l\, a) \sim_\beta t'$ with $\vdash_S ||t'||_\emptyset : f$. This means that t is β-equivalent to a term starting with four abstractions (otherwise $||t'||_\emptyset$ would be undefined) which implies $t \sim_\beta \lambda z \lambda s \lambda l \lambda a\, t'$ (because z, s, a, l are not free in t). Thus t is an internal Debruijn representation of a term u ($u = ||t'||_\emptyset$) with $\vdash_S u : f$. ■

4 Formalization of the semantics of minimal logic

Definition 10 *A model is given by a binary operation $\circ$, a constant e, and a predicate M satisfying the following conditions:*

- $\mathrm{M}_0(\circ) = \forall p,q,r\ p \circ (q \circ r) = (p \circ q) \circ r$ *($\circ$ is associative)*
- $\mathrm{M}_1(\circ, e) = \forall p\ e \circ p = p$
- $\mathrm{M}_2(\circ, M) = \forall f,g,p \begin{pmatrix} \forall q\ (\forall r\ M(r \circ (q \circ p), f) \rightarrow M(q \circ p, g)) \rightarrow \\ M(p, f \Rightarrow g) \end{pmatrix}$
- $\mathrm{M}_3(M) = \forall f,g,p\ (M(p, f \Rightarrow g) \rightarrow M(p, f) \rightarrow M(p, g))$

Such a model is very similar to a kripke tree model. Indeed, if we consider that a point in a tree is denoted by his path and that $\circ$ is the concatenation of paths in a tree, then our definition is equivalent to the usual one. We just replace $p > q$ by $p = r \circ q$.

The only difference is that we do not require the truth to increase when we climb in the tree. Instead, in M_2, we check that the formula is true at every point above $q \circ p$. The truth of f at p is not $M(p,f)$ but $\forall q\ M(q \circ p, f)$.

Definition 11 *The truth can be defined as being true in every model. This is formalized by the following predicate:*

$$\mathrm{Truth}(f) = \forall o\ \forall e\ \forall M \begin{pmatrix} \mathrm{M}_0(o) \rightarrow \mathrm{M}_1(o, e) \rightarrow \mathrm{M}_2(o, M) \rightarrow \mathrm{M}_3(M) \rightarrow \\ \forall p\ M(p, f) \end{pmatrix}$$

Definition 12 *We say that a λ-term t is an higher-order representation of a term u if t is closed, $t = \lambda l \lambda a\, t'$ and $||t'|| = u$ where $||t'||$ is the partial function taking a term as argument and defined by:*

- $||x|| = x$.
- $||(l\ \lambda x\, t_1)|| = \lambda x ||t_1||$.
- $||(l\ t_1)|| = \lambda x ||(t_1\ x)||$ *if t_1 does not start with λ and x not free in t_1.*
- $||(a\ t_1\ t_2)|| = (||t_1||\ ||t_2||)$.

This definition is well founded, because the number of occurrences of l or a decreases in each recursive call. Formally, $||t'||$ depends on the choice of the variables l and a but we omit these parameters in the notation.

We need the two cases for l, because we can extract terms containing sub-terms of the form $(l\ (a, t)) \sim_\eta (l\ \lambda x (a\, t\, x))$ from proofs of $\mathrm{Truth}(f)$.

Proposition 13 *For any term t and any formula f, if $\vdash_{AF_2} t : \mathrm{Truth}(f)$ then $(t\ (\lambda x\, x)\ (\lambda x\, x))$ is β-equivalent to an higher-order representation of a closed λ-term u.*

This proposition is weaker that the corresponding theorem 9 for provability. The stronger theorem that also states that u is of type f will be given later (theorem 26).

Proof: Let us call T the follwing formula:

$$T = \forall M \begin{pmatrix} \forall X\,(X \to X) \to \forall X\,(X \to X) \to \\ ((M \to M) \to M) \to (M \to M \to M) \to M \end{pmatrix}$$

This formula is obtained from Truth(f) by erasing all first-order information.

We choose t such that $\vdash_{AF_2} t : \text{Truth}(f)$. Therefore, we have $\vdash_{AF_2} t : T$ and using the adequation lemma, we have $t \in |T|^\sigma$ for any interpretation σ.

We choose two λ-variables a, l and an infinite set of variables $\{x_n\}_{n\in\mathbb{N}}$ not free in t. We define Φ the smallest set closed under β-equivalence such that:

- $x_n \in \Phi$ for all $n \in \mathbb{N}$
- $(l\,t_1) \in \Phi$ if $\forall n \in \mathbb{N}\,(t_1\,x_n) \in \Phi$
- $(a\,t_1\,t_2) \in \Phi$ if $t_1 \in \Phi$ and $t_2 \in \Phi$

We show, by induction on the definition of Φ, that $t \in \Phi$ and t normal imply that $||t||$ is defined (i) and the free variables of $||t||$ are among $\{a, b\} \cup \{x_n\}_{n\in\mathbb{N}}$ (ii). The only non trivial case is when $t = (l\,t_1)$. Then we choose x_n not free in t_1 and and we distinguish two cases:

- If t_1 does not start with λ, we have $(t_1\,x_n) \in \Phi$ and normal (by definition of Φ). By induction hypothesis, we get $||(t_1\,x_n)||$ defined and (ii) for $(t_1\,x_n)$. Therefore $||(l\,t_1)|| = \lambda x_n\,||(t_1\,x_n)||$ is defined and $(l\,t_1)$ satisfies (ii).
- If $t_1 = \lambda x\,t_2$ then $t_2[x/x_n]$ is normal and $t_2[x/x_n] \in \Phi$. Thus by induction hypothesis, we get $||t_2[x/x_n]||$ defined and (ii) for $t_2[x/x_n]$. Therefore $||(l\,t_1)|| = \lambda x\,||t_2|| = \lambda x_n\,||t_2[x/x_n]||$ is defined (because $||t_1[x/y]|| = ||t_1||[x/y]$ is immediate by induction on t_1) and $(l\,t_1)$ satisfies (ii).

We define the interpretation $\sigma' = \sigma[M/\Phi]$ and we prove the following properties:

- $\lambda x\,x \in |\forall X\,(X \to X)|^{\sigma'}$
- $l \in |(M \to M) \to M|^{\sigma'}$
- $a \in |M \to M \to M|^{\sigma'}$

The first condition is trivial and the last one uses the third case in the definition of Φ. For the second one, we choose $v \in \Phi \to \Phi$ and we must prove $(l\,v) \in \Phi$. For any $n \in \mathbb{N}$ we find $(v\,x_n) \in \Phi$ because $x_n \in \Phi$. Thus, we have $(l\,v) \in \Phi$.

Therefore, we have $t \in |T|^\sigma$ implies $(t)\lambda x\,x\ \ \lambda x\,x\ \ l\ \ a \in |M|^{\sigma'} = \Phi$. Thus, using (ii) we find $t' \sim_\beta (t\,(\lambda x\,x)\,(\lambda x\,x)\,l\,a)$ with $||t'||$ defined. From this we get $(t\,(\lambda x\,x)\,(\lambda x\,x)) \sim_\beta \lambda l \lambda a\,t'$ (because a, l are not free in t) with $\lambda l \lambda a\,t'$ closed (because the x_n variables were also chosen not free in t). We have what we wanted: $(t\,(\lambda x\,x)\,(\lambda x\,x))$ is an higher-order representation of $||t'||$. ■

Theorem 14 (Correctness of the semantics)

$$\forall f\ (\emptyset \Vdash f \to \mathrm{Truth}(f))$$

Proof: Together with the proof, we give the extracted term. We assume $\emptyset \Vdash f$ and we choose an arbitrary model defined by $\circ$, e, M. We must prove $M(p,f)$. We define the following predicate:

$$\mathrm{M_L}(p, C) = \forall q\ \mathbb{L}_{M(q \circ p)}(C)$$

$\mathrm{M_L}(p, C)$ means that C is a list of formulas true at any point above p. We easily prove the following lemmas about $\mathrm{M_L}$:

(1) $\forall p_0,f_0,c\ (\mathrm{M_L}(p_0,f_0 c) \to \forall q\ M(q \circ p_0,f_0))$
(2) $\forall p_0,f_0,c\ (\mathrm{M_L}(p_0,f_0 c) \to \mathrm{M_L}(p_0, c))$

The term `car` and `cdr` extracted respectively from (1) and (2) implement the usual destructors for lists (this is an easy consequence of the proposition 6 and the adequation lemma). Now to do the induction, we substitute the following predicate to the variable X in $\emptyset \Vdash f$:

$$\lambda C,f_0 \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(h(q,p_0),f_0))$$

Therefore, we will get the expected conclusion $M(p,f)$ if we prove the following properties:

- $\forall f_0, C, p_0\ (\mathrm{M_L}(p_0,f_0 C) \to \forall q\ M(q \circ p_0,f_0))$ which is (1).
- $\forall f_0, g, C \begin{pmatrix} \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(q \circ p_0,f_0)) \to \\ \forall p_0\ (\mathrm{M_L}(p_0, gC) \to \forall q\ M(q \circ p_0,f_0)) \end{pmatrix}$ which is immediate using (2). The extracted term is $\mathtt{weak_{cr}} = \lambda u \lambda c(u\,(\mathtt{cdr}\,c))$.
- $\forall f_0, g, C \begin{pmatrix} \forall p_0\ (\mathrm{M_L}(p_0,f_0 C) \to \forall q\ M(q \circ p_0, g)) \to \\ \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(q \circ p_0,f_0 \Rightarrow g)) \end{pmatrix}$ which is easy using $\forall p_0, q, r\ p_0 \circ (q \circ r) = (p_0 \circ q) \circ r$, $\forall p_0\ e \circ p_0 = p_0$ and $l : \mathrm{M_2}(\circ, M)$. The extracted term is $\mathtt{lam_{cr}}[l] = \lambda u \lambda c(l\,\lambda x(u\,(\mathtt{cons}\,x\,c)))$ where `cons` implements the expected operation on lists.
- $\forall f_0, g, C \begin{pmatrix} \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(q \circ p_0,f_0 \Rightarrow g)) \to \\ \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(q \circ p_0,f_0)) \to \\ \forall p_0\ (\mathrm{M_L}(p_0, C) \to \forall q\ M(q \circ p_0, g)) \end{pmatrix}$ which is easy using $\forall p_0, q, r\ p_0 \circ (q \circ r) = (p_0 \circ q) \circ r$, $\forall p_0\ e \circ p_0 = p_0$ and $a : \mathrm{M_3}(M)$. The extracted term is $\mathtt{app_{cr}}[a] = \lambda u \lambda v \lambda c(a\,(u\,c)\,(v\,c))$.
- $\mathrm{M_L}(p,\emptyset)$ which is trivial. The extracted term is $\mathtt{nil} = \lambda x \lambda f\, x$.

The term extracted from the all proof is

$$\mathtt{cr} = \lambda t \lambda e_1 \lambda e_2 \lambda l \lambda a (t\ \mathtt{car}\ \mathtt{weak_{cr}}\ \mathtt{lam_{cr}}[l]\ \mathtt{app_{cr}}[a]\ \mathtt{nil})$$

Lemma 15 *We define* $\sigma = [z/\mathtt{car}, s/\mathtt{weak_{cr}}, l/\mathtt{lam_{cr}}[l], a/\mathtt{app_{cr}}[a]]$ *and, for* $c = x_n \ldots x_1$, $\overline{c} = \lambda a \lambda f (f\, x_n\, (f\, x_{n-1} \ldots (f\, x_1\, a) \ldots))$. *We have* $||t'||_c = u$ *implies that* t'' *the normal form of* $(t'\sigma)\overline{c}$ *exists and* $||t''|| = u$.

Proof: We prove this by induction on the term t'. By definition of $||t'||_c$, we are in one of the following cases:

- If $t' = z$ then $(t'\sigma\,\overline{c}) \sim_\beta (\mathtt{car}\,\overline{c})$. As $||z||_c$ is defined, c starts with a variable x and $u = x$. We can take $t'' = x$.
- If $t' = (s\, t'_1)$ then $(t'\sigma\,\overline{c}) \sim_\beta (\mathtt{weak_{cr}}\, t'_1\sigma\,\overline{c}) \sim_\beta (t'_1\sigma\,(\mathtt{cdr}, \overline{c}))$. We have $c = xc'$ and $||t'||_c = ||t'_1||_{c'} = u$ which gives what we want using the induction hypothesis.
- If $t' = (l\, t'_1)$ then $(t'\sigma\,\overline{c}) \sim_\beta (\mathtt{lam_{cr}}[l]\, t'_1\sigma\,\overline{c}) \sim_\beta (l\, \lambda x (t'_1\sigma\,(Cons\, x\, \overline{c})))$. We have $||t'||_c = \lambda x ||t'_1||_{xc}$ and the induction hypothesis gives $(t'_1\sigma\,(Cons\, x\, \overline{c})) \sim_\beta t''_1$ normal with $||t''_1|| = ||t'_1||_{xc}$. Then, we take $t'' = (l\, \lambda x\, t''_1)$ and we get what we expected.
- If $t' = (a\, t'_1\, t'_2)$ then $(t'\sigma\,\overline{c}) \sim_\beta (\mathtt{app_{cr}}[a]\, t'_1\sigma\, t'_2\sigma\,\overline{c}) \sim_\beta (a\,(t'_1\sigma\,\overline{c})\,(t'_2\sigma\,\overline{c}))$. We have $||t'||_c = (||t'_1||_c)||t'_2||_c$ and the induction hypothesis gives $(t'_1\sigma)c \sim_\beta t''_1$ normal with $||t''_1|| = ||t'_1||_c$ and the same for t'_2. We take $t'' = (a\, t''_1\, t''_2)$ and we get what we wanted. ■

Theorem 16 *If* t *is* β*-quivalent to an internal Debruijn representation of a term* u *then* $(\mathtt{cr}\, t)$ *is* β*-equivalent to an higher-order representation of* u.

Proof: Let t be an internal Debruijn representation of a term u. We have $t \sim_\beta \lambda z \lambda s \lambda l \lambda a\, t'$ with $||t'||_\emptyset = u$. Then $(\mathtt{cr}\, t) \sim_\beta \lambda e_1 \lambda e_2 \lambda l \lambda a (t'\sigma\,\overline{\emptyset})$ where σ is the substitution defined in the proof of the previous lemma (because $\mathtt{nil} = \overline{\emptyset}$). From the definition of σ, we know that a and l are the only free variables of $(t'\sigma\,\overline{\emptyset})$ Therefore, using the previous lemma, $(\mathtt{cr}\ t\ (\lambda z\, z)\ (\lambda z\, z)) \sim_b eta\lambda l \lambda a (t'\sigma\,\overline{\emptyset})$ and the normal form t'' of $(t'\sigma\,\overline{\emptyset})$ exists and verifies $||t''|| = u$. ■

6 Completeness

Before proving the completeness of our semantics, we need a few definition and axioms:

Definition 17 *We define the type of untyped list by*

$$\mathbb{L}(l) = \forall X\ (X(\emptyset) \to \forall a\ \forall l'{:}X\ X(al') \to X(l))$$

Lemma 18 *If* $\vdash_{AF2} t : \mathbb{L}(l)$ *then* t *is* β*-equivalent to the church numeral* $\overline{n} = \lambda x \lambda f (f^n\, x)$ *where* n *is the length of the list* l*.*

Proof: The proof is exactly the same as the proof for the type of Church numeral itself (see [4]). ■

Definition 19 *To prove the completeness theorem, we need to add one binary function symbol* $l \mathbin{@} l'$ *for the concatenation of lists and a unary function symbol* $\mathrm{tail}(l)$ *for the function removing the last element in a list.*

Axiom 20 *We then assume the following axioms:*

- $\forall a,l,l'\ al \mathbin{@} l' = a(l \mathbin{@} l')$
- $\forall l\ \emptyset \mathbin{@} l = l$
- $\forall l\ l \mathbin{@} \emptyset = l$
- $\forall l,l',l''\ l \mathbin{@} (l' \mathbin{@} l'') = (l \mathbin{@} l') \mathbin{@} l''$
- $\mathrm{tail}(\emptyset) = \emptyset$
- $\forall a\ \mathrm{tail}(a\emptyset) = \emptyset$
- $\forall l,l',a\ \mathrm{tail}(l \mathbin{@} al') = l \mathbin{@} \mathrm{tail}(al')$

Remark: we could prove the completeness theorem without axioms if we added some extra conditions in the definition of models. But this would pollute the extracted terms.

Lemma 21 *We can prove the following:* $\forall l_1,l_2\ (\mathbb{L}(l_2 \mathbin{@} l_1) \to \mathbb{L}(l_1) \to \mathbb{L}(l_2))$

Proof: We first prove $\forall l{:}\mathbb{L}\ \mathbb{L}(\mathrm{tail}(l))$ by an easy induction on the structure of the list. Then $\forall l,a\ (\mathbb{L}(l \mathbin{@} a\emptyset) \to \mathbb{L}(l))$ (i) follows easily. To prove the lemma, we proceed by induction on the structure of the list l_1 using (i) for the cons case. ■

Lemma 22 *If* t *is a term extracted from a proof of lemma 21 then* t *computes the subtraction of two Church integers (when the first argument is greater or equal to the second one).*

Proof: This is an immediate consequence of the lemma 18. ■

Theorem 23 *The semantics is complete:*

$$\forall f{:}\mathrm{Truth}\ \emptyset \Vdash f$$

Proof: Together with the proof, we will construct the extracted term. Using the definition of $\Vdash$, we prove $X(\emptyset,f)$ using the following hypothesis:

(1) $x : \mathrm{Truth}(f)$
(2) $z : \forall f_0, C\ X(f_0\,C,f_0)$
(3) $s : \forall f_0, g, C\ (X(C,f_0) \to X(g\,C,f_0))$
(4) $l : \forall f_0, g, C\ (X(f_0\,C,g) \to X(C,f_0 \Rightarrow g))$
(5) $a : \forall f_0, g, C\ (X(C,f_0 \Rightarrow g) \to X(C,f_0) \to X(C,g))$

We replace o by the concatenation of lists, e by $\emptyset$ and M by $\lambda C,f_0(\mathbb{L}(C) \to X(C,f_0))$ in (1). This means that we consider the model whose points are contexts and where X defines the truth. Thus we have to prove the following:

- $\mathrm{M}_0(@)$ and $\mathrm{M}_1(@,\emptyset)$ are among the axioms we assumed. We can always consider that $\mathtt{idt} = \lambda x\,x$ is the algorithmic content of the equational axioms to extract a term from these axioms.
- To prove $\mathrm{M}_2(@,\lambda C,f_0(\mathbb{L}(C) \to X(C,f_0)))$, we assume
  $\cdot\ f : \forall q \begin{pmatrix} \forall r\ (\mathbb{L}(r\ @\ q\ @\ p) \to X(r\ @\ q\ @\ p,f_0)) \to \\ \mathbb{L}(q\ @\ p) \to X(q\ @\ p,g) \end{pmatrix}$ (i)
  $\cdot\ n : \mathbb{L}(p)$ (ii)
  and we must prove $X(p,f_0 \Rightarrow g)$. Using (4) and (i), we just need to prove $\forall r\ (\mathbb{L}(r\ @\ f_0\emptyset\ @\ p) \to X(r\ @\ f_0\emptyset\ @\ p,f_0))$ and $\mathbb{L}(f_0\emptyset\ @\ p)$. The second one is immediate from (ii). For the first one, we assume $p : \mathbb{L}(r\ @\ f_0\emptyset\ @\ p)$ and we will prove $X(r\ @\ f_0\emptyset\ @\ p,f_0)$. Using lemma21, $n : \mathbb{L}(p)$ and $p : \mathbb{L}(r\ @\ f_0\emptyset\ @\ p)$, we get $(\mathtt{sub}\,p\,(\mathtt{succ}\,n)) : \mathbb{L}(r)$ ($\mathtt{succ}$ is a term computing the successor of a Church numeral and by lemma 22, $\mathtt{sub}$ is a term computing the subtraction of two Church numerals). Then we prove $X(r\ @\ f_0\emptyset\ @\ p,f_0)$ by induction on r. For the nil case, we get $X(\emptyset\ @\ f_0\emptyset\ @\ p,f_0)$ using our equational axioms and (2). For the cons case, we get $X(l'\ @\ f_0\emptyset\ @\ p,f_0) \to X(a l'\ @\ f_0\emptyset\ @\ p,f_0)$ using (3). The term extracted from this sub-proof is

$$\mathtt{lam}_{\mathtt{cp}}[l] = \lambda f\lambda n\,(l\ (f\ (\lambda p\,(\mathtt{sub}\ p\ (\mathtt{succ}\,n)\ z\ s))\ (\mathtt{succ}\,n)))$$

- To prove $\mathrm{M}_3(\lambda C,f_0(\mathbb{L}(C) \to X(C,f_0)))$, we just need to use (5) and the extracted term is
$$\mathtt{app}_{\mathtt{cp}}[a] = \lambda y\lambda y'\lambda n(a\,(y\,n)\,(y'\,n))$$
- Finally, $\mathtt{nil} = \lambda a\lambda f\,a$ is extracted trivially from a proof of $\mathbb{L}(\emptyset)$.

The term extracted from the all proof is

$$\mathtt{cp} = \lambda k\lambda z\lambda s\lambda l\lambda a(k\ \mathtt{idt}\ \mathtt{idt}\ \mathtt{lam}_{\mathtt{cp}}[l]\ \mathtt{app}_{\mathtt{cp}}[a]\ \mathtt{nil})$$

Lemma 24 *We define* $\sigma_n = [l/\mathtt{lam}_{\mathtt{cp}}[l], a/\mathtt{app}_{\mathtt{cp}}[a], x_1/\mathtt{K}[\overline{1}], \ldots, x_n/\mathtt{K}[\overline{n}]]$ *where* $\mathtt{K}[q] = \lambda p(\mathtt{sub}\,p\,q\,z\,s)$. *If* $||t'|| = u$ *where* $x_1, \ldots x_n$ *are distinct and are the free variables of* u, *then* $||t''||_{x_n \ldots x_1} = u$ *with* t'' *the normal form of* $(t'\sigma_n)\overline{n}$.

Proof: We first notice that

$$(\mathtt{lam_{cp}}[l]\, v\, \overline{q}) \sim_\beta (l\, (v\, \mathtt{K}[\overline{q+1}]\, \overline{q+1})) \quad (i)$$

Then we prove the result by induction on u:

- If $u = x_q$, then $t' = x_q$ and $(t'\sigma_n\, \overline{n}) \sim_\beta (\mathtt{K}[q]\, \overline{n}) \sim_\beta (s^{n-q}\, z)$ and we get what we wanted by an easy induction on $n - q$.
- If $u = \lambda x_{n+1}\, u_1$, then $t' = (l\, \lambda x_{n+1}\, t'_1)$ with $||t'_1|| = u_1$. Using (i), we find $(t'\sigma_n\, \overline{n}) \sim_\beta (\mathtt{lam_{cp}}[l]\, (\lambda x_{n+1} t'_1 \sigma_n)\, \overline{n}) \sim_\beta (l\, t'_1 \sigma_{n+1}\, \overline{n+1}) \sim_\beta (l\, t''_1) = t''$ where t''_1 is the normal form of $(t'_1\sigma_{n+1})\overline{n+1}$. Using the induction hypothesis, we find $||t''||_{x_n \ldots x_1} = \lambda x_{n+1} ||t''_1||_{x_{n+1} \ldots x_1} = \lambda x_{n+1} u_1 = u$ which is what we wanted.
- If $u = (u_1\, u_2)$, then $t' = (a\, t'_1\, t'_2)$. We have $(t'\sigma_n\, \overline{n}) \sim_\beta (\mathtt{app_{cp}}[a]\, t'_1\, t'_2\, \overline{n}) \sim_\beta (a\, (t'_1\sigma_n\, \overline{n})\, (t'_2\sigma_n\, \overline{n}))$ and we easily get the expected result using the induction hypothesis and the definition of $||t''||_{x_n \ldots x_1}$. ■

Proposition 25 *If* $(t\, \mathtt{idt}\, \mathtt{idt})$ *is* β*-equivalent an higher-order representation of a term* u *then* $(\mathtt{cp}\, t)$ *is* β*-equivalent to an internal Debruijn representation* t'' *of* u.

Proof: By definition, we have $(t\, \mathtt{idt}\, \mathtt{idt}) \sim_\beta \lambda l \lambda a t'$ with $||t'|| = u$. We have $(\mathtt{cp}\, t) \sim_\beta (t'[l/\mathtt{lam_{cp}}[l], a/\mathtt{app_{cp}}[a]]\, \mathtt{nil}) = (t'\sigma_0\, \mathtt{nil})$. Using the previous lemma, we find $(\mathtt{cp}\, t) \sim_\beta t''$ with $||t''||_\emptyset = u$ which means that $(\mathtt{cp}\, t)$ is β-equivalent to an internal Debruijn representation t'' of u. ■

Theorem 26 *For any term* t *and any formula* f*, if* $\vdash_{AF_2} t : \mathrm{Truth}(f)$ *then* $(t)\lambda x\, x\ \lambda x\, x$ *is* β*-equivalent to the higher-order representation of a* λ*-term* u *with* $\vdash_S u : f$.

This theorem is a refinement of the proposition 13.

Proof: Let t be a term such that $\vdash_{AF_2} t : \mathrm{Truth}(f)$. Then by the proposition 13 $(t\, \mathtt{idt}\, \mathtt{idt})$ is β-equivalent to an higher-order representation of a term u. Using the proposition 25, we find that $(\mathtt{cr}\, t) \sim_\beta t''$ where t'' is an internal Debruijn representation t'' of a term u.

Moreover, using the proof of the theorem completeness, we find $\vdash_{AF_2} (\mathtt{cr}\, t) : \emptyset \Vdash f$, and using the theorem 9 we find that $(\mathtt{cr}\, t)$ is β-equivalent to an internal Debruijn representation t''_0 of a term u' with $\vdash_S u' : f$. But using the definition of internal Debruijn representation we have $t'' \sim_\beta t''_0$ implies $t'' = t''_0$ (because t'' and t''_0 are in normal form and β-equivalent to the same term) and therefore $u = u'$

Thus, $(\mathtt{cr}\, t)$ is β-equivalent to an internal Debruijn representation t'' of a term u with $\vdash_S u : f$. ■

Theorem 27 *Any term T extracted from a proof of the completeness (resp. correctness) theorem translates an higher-order (resp. internal Debruijn) representation of a term u such that $\vdash_S u : f$ into an internal Debruijn (resp. higher-order) representation of a term u' such that $\vdash_S u' : f$.*

Proof: This theorem is an immediate consequence of the theorems 9 and 26 using the following properties (which are the converse of these theorems):

(1) If t is an internal Debruijn representation of u and $\vdash_S u : f$ then $\vdash_{AF_2} t : \emptyset \Vdash f$. We prove this by proving by induction on the proof that $||t||_{x_n \ldots x_1} = u$ and $x_1 : f_1, \ldots, x_n : f_n \vdash_S u : f$ implies $\vdash_{AF_2} t : f_n \ldots f_1 \Vdash f$. The proof is easy using the definition of $||t||_{x_n \ldots x_1}$.

(2) If t is an higher-order representation of u and $\vdash_S u : f$ then $\vdash_{AF_2} t :$ Truth(f). This result is a consequence of (1), the correctness theorem and the proposition 16. ■

References

[1] A. Heyting. *Les Fondements des Mathématiques. Intuitionisme. Théorie de la Démonstration.* Gauthier-Villars, Paris and E. Nauwelaersts, Louvain, 1955.

[2] W. Howard. The formulae-as-types notion of construction. *To H.B. Curry: Essays on combinatory logic, λ-calculus and formalism*, pages 479–490, 1980.

[3] S. A. Kripke. Semantical analysis of intuitionistic logic i. In *Formal system and recursive functions*, Oxford, 1965. North-Holland. 8th Logic colloquium.

[4] J.-L. Krivine. *Lambda-Calculus : Types and Models.* Computers and their applications. Ellis Horwood, 1993.

[5] J.-L. Krivine and Michel Parigot. Programming with proofs. *Inf. Process. Cybern.*, EIK 26(3):149–167, 1990.

[6] Jean-Louis Krivine. Une preuve formelle et intuitionniste du théorème de complétude de la logique classique. *Bulletin of Symbolic Logic*, 2(4):405–421, 1996.