System ST
β-reduction and completeness

Christophe Raffalli
LAMA, Université de Savoie
73376 Le Bourget-du-Lac cedex (FRANCE)
raffalli@univ-savoie.fr

Abstract

We prove that system ST (introduced in a previous work) enjoys subject reduction and is complete for realizability semantics. As far as the author knows, this is the only type system enjoying the second property.

System ST is a very expressive type system, whose principle is to use two kinds of formulae: types (formulae with algorithmic content) and propositions (formulae without algorithmic content). The fact that subtyping is used to build propositions and that propositions can be used in types trough a special implication gives its great expressive power to the system: all the operators you can imagine are definable (union, intersection, singleton, . . .).

Keywords: lambda-calcul, type, subtype

1 Introduction.

Motivation: System ST (introduced in [16]) is based on higher-order logic, using two sorts of formulae:

- Types: formulae with algorithmic content (denoted by the letters $A, B, C, \ldots$)
- Propositions: formulae without algorithmic content (denoted by the letters $P, Q, \ldots$).

This idea is already present in the work of Christine Paulin-Mohring for CoQ [12] and the work of Nora Szasz and Paula Severi for Martin-Löf's Type Theory [18]. Both sorts of formulae can use implication and universal quantification and they are related using two connectives:

- Subtyping: If A and B are types then, $A \subset B$ is a proposition meaning that any program of type A also inhabits type B.
- Special implication: If P is a proposition and if A is a type then, $P \Rightarrow\!\!\!\!\!\Rightarrow A$ represents A if P is true and the type containing all terms otherwise. This special implication seems to have been first introduced by the author in [16] and is central to system ST.

Using this simple idea, the natural rules for these connectives and a few axioms, we have shown in [16] that system ST can be used to extract programs from proofs, but also to prove programs. For instance, we typed a fixpoint operator with a type corresponding to well founded induction (using no specific rule) and we proved that the type of Church's numerals in system F is only inhabited with Church's numerals, internally in system ST (usually, this result can not be expressed inside the type system itself).

However, there were two open questions in our previous paper:

- We could not find logical axioms or rules to obtain the subject reduction property for β-reduction. We found partial answers, but it was not clear if β-reduction could be finitely axiomatizable in the system.
- Is realizability semantics complete? This property is never true for type systems. But, the main originality of system ST is that singleton types are definable (that is, for each λ-term, there is a type containing only that λ-term, up to $\beta\eta$-equivalence). This, and the fact that usually untypable terms (such as fixpoint operators) could be typed in system ST, made us think that the system could be complete.

In fact, we show in this paper that the answers to both questions are related and is "yes", if we add three axioms and two rules. We also show that system ST as the subject reduction property for the $\beta\eta$-equivalence. Moreover, all these axioms and rules have a clear meaning not directly related to the property we want to prove. For instance, the axiom specifically related to the subject reduction for β says that each type as a complement. About the last three axioms, which are quite complex, we will at least prove we cannot replace them by a proper subset.

Why do we need yet another type system ? By our completeness theorem, system ST is equivalent to realizability. We think it is better: it is a type system with a very rich language to write types. Therefore, it is much more natural to do proofs in system ST than in realizability (in [16] we give a proof for the fixpoint rule of TTR [11] without ordinal and the author found the proof using system ST while he could not find it using realizability). We have a plan to develop a computer language with annotations in system ST to prove the correctness of programs (ideas similar to [10, 13]). The language of the type system should allow an elegant and intuitive way to annotate programs with a lot of freedom for the programmer.

However, it should be clear that this article and [16] introduce system ST by studying its theoretical properties. The examples and the completeness theorem allow us to think that system ST could have practical interest, but this is yet to establish in future work.

System ST can also be seen as an elegant way to present realizability (which is not so easy to formalize, see the remark on the formalization in the section 8) with a possible stratification depending on the axioms or rules you choose! This may leads to interesting theoretical results ?

Related works on subtyping: there has been many works on subtyping [9] (see [7, 15] for a long but still incomplete list of references). However, in most (if not all ?) of these works, subtyping relations do not appear in types or appear through bounded quantifications. This means that all subtyping hypotheses in judgment are of the form $X \subset A$ or $A \subset X$. In our system, subtyping hypotheses are not restricted and subtyping can appear anywhere in types.

Moreover, you can usually deduce typing judgments from subtyping judgments, but never (as far as the author knows) in the other direction. In fact, all previous works on subtyping restrict the expressive power you could have to keep some properties (decidability of the subtyping relation, subject reduction, strong or weak normalization, ...).

In system ST, we kept as most freedom as possible, loosing decidability and normalization, but keeping subject reduction and gaining completeness. Indeed, previous systems enjoy often some kind of completeness for subtyping [9], but never the completeness of typing.

Related works on programs extraction. The initial goal of this work was to develop a type system, with nice properties, allowing the removal of parts of proofs with no algorithmic content when extracting programs. Works in this setting already exists [2, 4], but they start from a proof, for instance in Girard's system F [5], and try to discover the computationally useless parts of this proof.

Here we tried to develop a type system that was intended to be used directly to build proofs. Therefore, we are free to design a powerful and elegant system with no care about the decidability of some particular problems.

Moreover, in [2, 12, 4, 18] subtyping is absent or is an external property over types while here it is central.

Other related works. Some operators like union, intersections and singleton have been studied for instance in [1, 3, 6, 14]. However, the purpose of system ST is not to study these operators and it is hard to compare these works with our paper. Indeed, we are mainly interested in completeness while these works focus on more standard properties like normalization or on practical applications, which is still a work to be done for system ST.

Nevertheless, when these systems type pure λ-terms, they are always subsystems of system ST (this claim is almost a consequence of our completeness theorem). At least, all the rules or axioms encountered by the author (for pure λ-calculus) are derivable or admissible in system ST (the intersection rule of system D is only admissible).

Plan: in section 2 and 3, we recall the definition of system ST (extended with two new rules and three axioms). Then, (section 4) we recall the results of [16]. In section 5, we introduce some new defined operators and their derived rules. The main tool used in this paper are singletons and we study, in section 6, three ways to write the property "being a singleton" and we show that they are equivalent. In section 7, we prove that system ST enjoys subject reduction for β. We present our semantics in section 8 and its completeness in section 9.

PhoX: all the proofs inside system ST have been conceived with PhoX [17] and are available from Internet.

2 The Syntax.

Definition 1 (sorts) A sort is either τ, the sort of types, o the sort of propositions or $s \to s'$ the function sort between two sorts of smaller size.

Definition 2 (expressions) The set of expressions is the set of simply typed λ-terms, using sorts as simple types, written using the following constants of given sort:

- $\Rightarrow_\epsilon : \epsilon \to \epsilon \to \epsilon$ for $\epsilon \in \{o, \tau\}$
- $\Rrightarrow : o \to \tau \to \tau$
- $\subset : \tau \to \tau \to o$
- $\forall^s_\epsilon : (s \to \epsilon) \to \epsilon$ for $\epsilon \in \{o, \tau\}$ and any sort s.

We consider β-equivalent expressions to be equal.

Remark: later we will use pure λ-calculus. The expressions introduced above are the logical part of the system

and despite the fact these are simply typed λ-terms, they are distinct from the pure λ-terms that will be associated with proofs. In short, objects on the left of the ":" sign will be pure λ-terms and objects on the right will be expressions.

Notation issues: We assume that variables always carry their sort. We will write only x when the sort of x is imposed by the context. Otherwise we write variables x^s when their sort is s.

To simplify, we will often write $\forall$ for $\forall^s_\epsilon$ and $\Rightarrow$ for $\Rightarrow_\epsilon$ for any value of ϵ or s when the context makes it possible to recover the missing information. We will also write $\forall x\, A$ instead of $\forall(\lambda x\, A)$ and we will use the symbol $\Rightarrow$, $\subset$ and $\Rrightarrow$ in infix notation (we write $A \subset B$ instead of $\subset A\, B$).

To make it easier to read formulae, we adopt the convention that the letters $A, B, C, \ldots, H$ represent types and the letters $P, Q, R, \ldots, W$ represent propositions. We will use these letters both for expressions and variables. The letters X, Y, Z will be used for variables representing types. The letters x, y, z will be used either for λ-variables (in pure λ-terms, on the left of the ":" sign) or for polymorphic variables in expressions (that is variables that can have any sort). In the same way, t, u, v will either denote a pure λ-term or a polymorphic expression.

To limit the number of parentheses, we will consider that our implications are right associative with equal priorities, and that quantification has the highest priority : $A \Rightarrow B \Rightarrow C$ means $A \Rightarrow (B \Rightarrow C)$, $A \Rightarrow P \Rrightarrow C$ means $A \Rightarrow (P \Rrightarrow C)$ and $\forall x\, A(x) \subset B$ means $(\forall x\, A(x)) \subset B$.

In the last two sections of the paper, we will sometimes use the standard mathematical notation $A(t_1, \ldots, t_n)$ for applications.

Definition 3 (context) A context is a set composed of propositions P and pairs of the form $x : A$, where A is a type and x is a λ-variable. Moreover, each λ-variable must be declared at most once in a context.

Example 4 The set $x : A, y : B, P, Q, R$ is a context if A and B have the sort τ and P, Q, R have the sort o.

Definition 5 (sequent) There are two kinds of sequents: $\Gamma \vdash t : A$ and $\Gamma \vdash P$ where Γ is a context, t is a λ-term, A is a type and P is a proposition.

3 The rules and axioms.

Axiom rules:

$$\frac{}{\Gamma, x : A \vdash x : A} \qquad \frac{}{\Gamma, P \vdash P}$$

Subtyping rules:

$$\frac{}{\Gamma \vdash A \subset A} \qquad \frac{\Gamma \vdash t : A \quad \Gamma \vdash A \subset B}{\Gamma \vdash t : B}$$

$$\frac{\Gamma \vdash A \subset B \quad \Gamma \vdash B \subset C}{\Gamma \vdash A \subset C}$$

Implication rules for types:

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x\, t : A \Rightarrow B} \qquad \frac{\Gamma \vdash t : A \Rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash (t\, u) : B}$$

$$\frac{\Gamma \vdash B \subset A \quad \Gamma \vdash A' \subset B'}{\Gamma \vdash A \Rightarrow A' \subset B \Rightarrow B'}$$

Implication rules for propositions:

$$\frac{\Gamma, P \vdash Q}{\Gamma \vdash P \Rightarrow Q} \qquad \frac{\Gamma \vdash P \Rightarrow Q \quad \Gamma \vdash P}{\Gamma \vdash Q}$$

Quantification rules for types:

$$\frac{\Gamma \vdash t : A(y)}{\Gamma \vdash t : \forall x\, A(x)}\ \dagger \qquad \frac{\Gamma \vdash t : \forall x\, A(x)}{\Gamma \vdash t : A(v)}\ \ddagger$$

$$\frac{\Gamma \vdash A \subset B(y)}{\Gamma \vdash A \subset \forall x\, B(x)}\ \dagger \qquad \frac{\Gamma \vdash A(v) \subset B}{\Gamma \vdash \forall x\, A(x) \subset B}\ \ddagger$$

Quantification rules for propositions:

$$\frac{\Gamma \vdash P(y)}{\Gamma \vdash \forall x\, P(x)}\ \dagger \qquad \frac{\Gamma \vdash \forall x\, P(x)}{\Gamma \vdash P(v)}\ \ddagger$$

$\dagger$: y is a variable having the same sort than x and not free in the conclusion of the rule.
$\ddagger$: v is an expression having the same sort than x.

Special implication rules:

$$\frac{\Gamma, P \vdash t : A}{\Gamma \vdash t : P \Rrightarrow A} \qquad \frac{\Gamma \vdash t : P \Rrightarrow A \quad \Gamma \vdash P}{\Gamma \vdash t : A}$$

$$\frac{\Gamma, P \vdash A \subset B}{\Gamma \vdash A \subset P \Rrightarrow B} \qquad \frac{\Gamma \vdash A \subset B \quad \Gamma \vdash P}{\Gamma \vdash P \Rrightarrow A \subset B}$$

Defined operators:

- $\bot_\tau := \forall^\tau_\tau X\, X$ False (empty) type
- $\bot_o := \forall^\tau_o X\, \forall^\tau_o Y\, (X \subset Y)$ False proposition
- $\top_\tau := \forall^\tau_\tau K (\forall^\tau_o X (X \subset K) \Rrightarrow K)$ True (Ω) type
- $\top_o := \forall^\tau_o X\, (X \subset X)$ True proposition
- $A \restriction P := \forall K\, ((A \subset P \Rrightarrow K) \Rrightarrow K)$ Parigot's restriction operator [11]

- $\bigcup x\ A(x) := \forall K\ (\forall x\ (A(x) \subset K) \Rrightarrow K)$ Union
- $A[B] := \forall X\ ((A \subset B \Rightarrow X) \Rrightarrow X)$ Direct image
- $A^{-1}[B] := \bigcup X\ (X \restriction (A \subset X \Rightarrow B))$ Inverse image
- $\Lambda X\ A(X) := \forall X\ (X \Rightarrow A(X))$ Lambda abstraction
- $A \cap B := \bigcup X\ (X \restriction ((X \subset A) \wedge (X \subset B)))$ Binary intersection
- $A \cup B := \forall K\ ((A \subset K) \Rrightarrow (B \subset K) \Rrightarrow K)$ Binary union
- $P \wedge Q := \forall K\ ((P \Rightarrow Q \Rightarrow K) \Rightarrow K)$ conjunction
- $P \vee Q := \forall K\ ((P \Rightarrow K) \Rightarrow (Q \Rightarrow K) \Rightarrow K)$ disjunction
- $P \Leftrightarrow Q := (P \Rightarrow Q) \wedge (Q \Rightarrow P)$ equivalence
- $\exists x\ P(x) := \forall K\ (\forall x\ (P(x) \Rightarrow K) \Rightarrow K)$ existential
- $A = \emptyset := A \subset \bot_\tau$ emptiness
- $A \neq \emptyset := A = \emptyset \Rightarrow \bot_o$ non-emptiness
- $A = B := (A \subset B) \wedge (B \subset A)$ equality on types
- $A^c := \bigcup X\ (X \restriction (X \cap A \subset \bot_\tau))$ type complement
- $A \setminus B := A \cap B^c$ types difference
- $\mathcal{S}_\tau(A) := A \neq \emptyset \wedge \forall B \left(\begin{array}{c} \left(A \subset \bigcup X^\tau\ B(X^\tau)\right) \Rightarrow \\ \exists X^\tau\ (A \subset B(X^\tau)) \end{array} \right)$ A is a singleton

  In this definition, X^τ has the sort τ (therefore, B has the sort $\tau \to \tau$). We will see that this definition implies the other definitions with B of sort $\sigma \to \tau$ for any σ.

Remark: the first eleven definitions where introduced in [16]. Most of these definitions (new or old) define natural operators and using the informal (or formal) semantics of the system, it is easy to check that they have the intended meaning. For instance, $\top_\tau$ is defined as the intersection of all types K containing any type X and this intersection only ranges over the type of all terms.

However, some definitions need a comment:

- Parigot's restriction operator ($A \restriction P$) is a conjunction between a type and a proposition. Its interpretation is A if P is true and the empty type otherwise.

  Moreover, used together with union, this operator allows to write the union of all types verifying a given property P (by writing $\bigcup X\ (X \restriction P(X))$).

- The direct and inverse image ($A[B]$ and $A^{-1}[B]$) are very surprising: the first one is the set of all terms obtained by applying a term in A to a term in B and it is not definable in any existing type system.

  The second one is the set of all terms u such that for any term t in A, $(t\,u)$ is in B. This is some kind of dual of the implication.

- The lambda abstraction ($\Lambda X\ A(X)$), used together with direct image, will allow us to write singleton types.

- The definition of "being a singleton" ($\mathcal{S}_\tau$) is clear but is not the most natural one: another definition would be minimal non empty element for the subtyping relation. But, there are semantics where this definition is too strong (not enough singletons).

Extra rules:

$$\frac{\Gamma \vdash t : (P \Rrightarrow \bot_\tau) \Rightarrow \bot_\tau}{\Gamma \vdash P} \qquad \frac{\Gamma, x : A \vdash A \subset B}{\Gamma \vdash A \subset B}$$

Left rules:

$$\frac{\Gamma, x : A \vdash t : B \quad \Gamma \vdash A' \subset A}{\Gamma, x : A' \vdash t : B} \qquad \frac{\Gamma, x : A(y) \vdash t : B}{\Gamma, x : \bigcup x\ A(x) \vdash t : B}\ \dagger$$

Remark: The two first rules where introduced in [16]. The two left rules are new in this paper. They seem to be needed to get the subject reduction property for β and our completeness theorems. These rules are derivable, but they introduce a β expansion.

Axioms: The rules given above were easy to understand. However, the following axioms are more complex. We will give a small comment explaining their intuitive meaning. For some of them, more explanation will be found in the sections about semantics or completeness.

- Mitchell's axiom [9]
  $\vdash \forall A,B\ (\forall x\ (A(x) \Rightarrow B(x)) \subset \forall x\ A(x) \Rightarrow \forall x\ B(x))$
- Inversion of implications
  $\vdash \forall P\ \forall A,B\ (P \Rrightarrow A \Rightarrow B \subset A \Rightarrow P \Rrightarrow B)$
  The two first axioms are similar. There are two ways to interpret them: technically, they are needed to save one η-expansion in terms; semantically, they express the fact that $\forall$ and $\Rrightarrow$ have no algorithmic contents.
- False is empty
  $\vdash \forall A,B\ (A \subset \bot_\tau \Rightarrow B)$
  A sufficient condition for this axiom to be true is that $\bot_\tau$ is interpreted by the empty type.
- Union axiom
  $\vdash \forall F\ \forall A\ (\forall x\ (F(x) \Rightarrow A) \subset \bigcup x\ F(x) \Rightarrow A)$
  A sufficient condition for this axiom to be true is that the union is really interpreted by a union.
- Atomicity
  $\vdash \forall A\ (A \subset \bigcup X\ (X \restriction (\mathcal{S}_\tau(X) \wedge (X \subset A))))$
  This axiom expresses that any type is the union of its singletons.

- Extensionality
  $\vdash \forall A,B\ (\mathcal{S}_\tau(B) \Rightarrow \forall X\ (A[X] \subset B[X]) \Rightarrow (A \subset B))$
  This axiom is related to η-equivalence. The hypothesis that B is a singleton is needed: if B is interpreted by $\{\lambda x(u\,u) | x \text{ not free in } u\}$ and A by $\{\lambda x(x\,x)\}$ then $\forall X\ (A[X] \subset B[X])$ is true and $A \subset B$ is not.
- Commutation of singletons
  $$\vdash \forall X\ \forall B \left(\mathcal{S}_\tau(X) \Rightarrow \left(\begin{array}{l} X \Rightarrow \bigcup Y^\tau\ B(Y^\tau) \subset \\ \qquad \bigcup Y^\tau\ (X \Rightarrow B(Y^\tau)) \end{array} \right) \right)$$
  This axiom is equivalent to the property "the direct image of a singleton by a singleton is a singleton" (see lemma 30). It is also related to the subject reduction property for β-expansion and not only head-expansion.
- Complement
  $\vdash \forall A,B\ (A \subset B \cup B^c)$
  This axiom is related to the subject reduction for β and may be seen as a form of excluded middle.

All axioms except the last three were introduced in [16]. The description of system ST ends here and no further axioms of rules will be added in this paper.

4 Previous results.

We give now the results proved in [16] that we use here.

Fact 6 *Both definitions of $False$ are equivalent. The implication in one direction can be expressed by the formula $\bot_o \nRightarrow \bot_\tau$. The converse implication can only be written as a derived rule:*

$$\frac{\Gamma \vdash t : \bot_\tau}{\Gamma \vdash \bot_o}$$

Fact 7 *We can prove $\vdash\ \forall P\ (((P \Rightarrow \bot_o) \Rightarrow \bot_o) \Rightarrow P)$ and derive the following rules of excluded middle:*

$$\frac{\Gamma, P \vdash Q \quad \Gamma, P \Rightarrow \bot_o \vdash Q}{\Gamma \vdash Q}$$

$$\frac{\Gamma, x : A \vdash Q \quad \Gamma, A \subset \bot_\tau \vdash Q}{\Gamma \vdash Q}$$

Fact 8 *We derive these rules for the restriction operator:*

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash P}{\Gamma \vdash t : A \restriction P} \qquad \frac{\Gamma \vdash t : A \restriction P}{\Gamma \vdash t : A} \qquad \frac{\Gamma \vdash t : A \restriction P}{\Gamma \vdash P}$$

$$\frac{\Gamma \vdash A \subset B \quad \Gamma \vdash P}{\Gamma \vdash A \subset B \restriction P} \qquad \frac{\Gamma, P \vdash A \subset B}{\Gamma \vdash A \restriction P \subset B}$$

Fact 9 *We prove the following statement:*

$$\forall P\ \forall A,B\ (P \nRightarrow A \Rightarrow B \subset (A \restriction P) \Rightarrow B).$$

Fact 10 Union rules *We derive these rules for union:*

$$\frac{\Gamma \vdash t : F(v)}{\Gamma \vdash t : \bigcup x\ F(x)}\ \ddagger$$

$$\frac{\Gamma \vdash t : \bigcup x\ F(x) \quad \Gamma \vdash u : F(y) \Rightarrow A}{\Gamma \vdash (u\,t) : A}\ \dagger$$

$$\frac{\Gamma \vdash A \subset F(v)}{\Gamma \vdash A \subset \bigcup x\ F(x)}\ \ddagger \qquad \frac{\Gamma \vdash F(y) \subset A}{\Gamma \vdash \bigcup x\ F(x) \subset A}\ \dagger$$

Fact 11 *We can derive the following rule and theorems for direct and inverse image:*

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : B}{\Gamma \vdash (t\,u) : A[B]}$$

$\vdash \forall A,B,C\ ((A \subset B \Rightarrow C) \Leftrightarrow (A[B] \subset C))$.
$\vdash \forall A,B,C\ ((A \subset B \Rightarrow C) \Leftrightarrow (B \subset A^{-1}[C]))$.

Definition 12 For every λ-term t and every mapping ϕ from λ-variables to formulae of sort τ, we define $|t|^\phi$ by induction as:

- $|x|^\phi := \phi(x)$.
- $|(t\,u)|^\phi := |t|^\phi[|u|^\phi]$.
- $|\lambda x\,t|^\phi := \Lambda X |t|^{\phi[x=X]} = \forall X(X \Rightarrow |t|^{\phi[x=X]})$.

Fact 13 *For any λ-term t, if $\{x_1, \ldots, x_n\}$ are the free variables of t, we prove $x_1 : \phi(x_1), \ldots, x_n : \phi(x_n) \vdash t : |t|^\phi$.*

Theorem 14 *Let Γ be $x_1 : A_1, \ldots, x_n : A_n, P_1, \ldots, P_q$ and ϕ defined by $\phi(x_i) = A_i$ for $i \in \{1, \ldots, n\}$. We can derive $\Gamma \vdash t : A$ if and only if we can derive $\Gamma \vdash |t|^\phi \subset A$.*

Theorem 15 *Our system has the subject reduction property for η and the subject expansion property for β. These results are immediate consequences of the previous theorem 14, fact 13 and the following provable subtyping relations:*

- $\vdash \forall A\ (A \subset \Lambda X\ A[X])$ *η-reduction*
- $\vdash \forall A\ \forall B\ ((\Lambda X\ A(X))[B] \subset A(B))$ *β-expansion*

Remark: it is important to note that the direction of the subtyping relation is inverted compared to the direction of the relation in the subject reduction property. Indeed, if $t \succ_\eta t'$, to deduce $t' : A$ from $t : A$, we use $|t'|^\phi \subset |t|^\phi \subset A$.

5 New derived rules.

Fact 16 *It is trivial to derive all the usual rules and properties of disjunction, conjunction and existential quantifiers on propositions.*

Proof The definitions are the usual ones used in second or higher order logic. ∎

Fact 17 direct image and abstraction are increasing

$$\vdash \forall A,A',B,B'\ ((A \subset A') \Rightarrow (B \subset B') \Rightarrow (A[B] \subset A'[B']))$$

$$\vdash \forall A,A'\ (\forall X\ (A(X) \subset A'(X)) \Rightarrow (\Lambda X\ A(X) \subset \Lambda X\ A'(X)))$$

Proof Immediate. ∎

Fact 18 *We can derive the following rule:*

$$\frac{\Gamma, x : A \vdash P}{\Gamma, A \neq \emptyset \vdash P}$$

Proof Consequence of the second rule in fact 7. ∎

Fact 19 Direct image and union

$$\vdash \forall A\ \forall B\ (\bigcup x\ A[B(x)] \subset A[\bigcup x\ B(x)])$$

$$\vdash \forall A\ \forall B\ (A[\bigcup x\ B(x)] \subset \bigcup x\ A[B(x)])$$

Proof The first subtyping is trivial from fact 17. The second one is a consequence of the fact 11. ∎

Fact 20 *We can prove the following inversion between intersection and restriction:*

$$\forall A,B\ \forall P\ ((A \restriction P) \cap B \subset (A \cap B) \restriction P)$$

Proof Easy, distinguishing the case where P is true and the case where P is false. ∎

Fact 21 *We give another definition of binary union:* $A \hat{\cup} B = \bigcup X\ (X \restriction (X = A \vee X = B))$ *and prove the equality:* $\forall A,B\ (A \cup B) = (A \hat{\cup} B)$

Proof Immediate. ∎

6 Singletons.

Definition 22 We give the following alternative definitions of singleton:

- $\mathcal{S}_\sigma(A) = \begin{pmatrix} A \neq \emptyset \wedge \\ \forall B \left(\begin{pmatrix} A \subset \bigcup x^\sigma\ B(x^\sigma) \end{pmatrix} \Rightarrow \\ \exists x^\sigma\ (A \subset B(x^\sigma)) \end{pmatrix} \right)$

  (for any sort σ)

- $\mathcal{S}_\cup(A) = \begin{pmatrix} A \neq \emptyset \wedge \\ \forall X,Y \left(\begin{pmatrix} (A \subset X \cup Y) \Rightarrow \\ (A \subset X) \vee \\ (A \subset Y) \end{pmatrix} \right) \end{pmatrix}$

- $\mathcal{S}_\subset(A) = \left(A \neq \emptyset \wedge \forall X \left(\begin{pmatrix} (X \subset A) \Rightarrow X \neq \emptyset \Rightarrow \\ (A \subset X) \end{pmatrix} \right) \right)$

Lemma 23 *For any sort σ we have*

$$\forall A\ (\mathcal{S}_\tau(A) \Rightarrow \mathcal{S}_\sigma(A)).$$

Proof Using the fact that
$\bigcup x^\sigma\ B(x^\sigma) \subset \bigcup Y^\tau\ (Y^\tau \restriction \exists x^\sigma\ (Y^\tau \subset B(x^\sigma)))$. ∎

Lemma 24 *We have*

$$\forall B\ \forall A\ (\bigcup x\ B(x) \cap A \subset \bigcup x\ (B(x) \cap A)).$$

Proof Consequence of the atomicity axiom and lemma 23. ∎

Fact 25 *We can prove the distributivity of binary union:*

$$\forall A,B,C\ ((B \cup C) \cap A \subset (B \cap A) \cup (C \cap A))$$

Proof Consequence of the lemma 24, fact 21 and 20. ∎

Fact 26 *We can prove* $\forall A,B\ (A \subset B \cup (A \setminus B))$.

Proof Easy using fact 25 and complement axiom. ∎

Lemma 27 *The definitions $\mathcal{S}_\tau$, $\mathcal{S}_\cup$ and $\mathcal{S}_\subset$ are equivalent.*

Proof

- $\forall A\ (\mathcal{S}_\tau(A) \Rightarrow \mathcal{S}_\cup(A))$: we assume $\mathcal{S}_\tau(A)$, $A \subset X \cup Y$ and prove $(A \subset X) \vee (A \subset Y)$. We get $A \subset X \hat{\cup} Y$ by fact 21. We apply the hypothesis $\mathcal{S}_\tau(A)$ with $A \subset X \hat{\cup} Y$ and find Z such that get $A \subset Z \restriction (Z = X \vee Z = Y)$. Using fact 7, we distinguish the case where $A \subset \bot_\tau$ and the case where $x : A$. The first case is trivial. In the second case, using fact 8, we get $A \subset Z$, $Z = X \vee Z = Y$ and we end the proof distinguishing the two cases.

- $\forall A\ (\mathcal{S}_\subset(A) \Rightarrow \mathcal{S}_\tau(A))$: We assume $\mathcal{S}_\subset(A)$ and $A \subset \bigcup X\ B(X)$ to prove $\exists X\ (A \subset B(X))$. Using classical reasoning (fact 7), we can assume $\forall X\ ((A \subset B(X)) \Rightarrow \bot_o)$ (i).

  We now prove $\forall X\ (B(X) \cap A \subset \bot_\tau)$ (ii). Using the second rule in fact 7, we can assume $x : B(X) \cap A$. Then, we get $A \subset B(X) \cap A$ from the hypothesis $\mathcal{S}_\subset(A)$ by proving $B(X) \cap A \subset A$ which is trivial and $B(X) \cap A \neq \emptyset$ which is an immediate consequence of $x : B(X) \cap A$. Finally, we get $B(X) \cap A \subset \bot_\tau$ trivially from (i) and $A \subset B(X) \cap A$.

  To end the proof, we use $A \subset \bigcup X\ B(X)$ and the lemma 24 to prove $A \subset \bigcup X\ (B(X) \cap A)$ (iii) and we get a contradiction between $A \neq \emptyset$, (ii) and (iii).

- $\forall A\ (\mathcal{S}_\cup(A) \Rightarrow \mathcal{S}_\subset(A))$: We assume $\mathcal{S}_\cup(A)$, $X \subset A$, $X \neq \emptyset$ and we prove $A \subset X$. We define $\mathrm{B} := A \setminus X$. From the fact 26, we have $A \subset X \cup \mathrm{B}$. Thus, using hypothesis $\mathcal{S}_\cup(A)$, we can assume $(A \subset X) \vee (A \subset \mathrm{B})$ to prove $A \subset X$. Therefore, we can assume $A \subset \mathrm{B}$.

  Then, we prove $X = \emptyset$ (which gives a contradiction) from $X \subset X^c$ which is a consequence of the hypothesis $X \subset A$ and $A \subset \mathrm{B}$ with the fact that $\mathrm{B} = (A \setminus X)$, using lemma 24 and 20. ∎

To prove proposition 31 saying that abstraction (Λ) preserves the property of begin a singleton, we need the following lemma:

Lemma 28 *We can give yet another definition of singleton:*

$$\mathcal{S}'_\subset(A) = \left(A' \neq \emptyset \wedge \forall X \begin{pmatrix} \mathcal{S}_\tau(X) \Rightarrow (X \subset A') \Rightarrow \\ X \neq \emptyset \Rightarrow (A' \subset X) \end{pmatrix}\right)$$

Proof We prove $\forall A\ (\mathcal{S}_\subset(A) \Leftrightarrow \mathcal{S}'_\subset(A))$. The left to right implication is trivial. For the other direction, we just need to prove $\forall X\ (X \neq \emptyset \Rightarrow \exists Y\ (\mathcal{S}_\tau(Y) \wedge (Y \subset X)))$ which is easy using the atomicity axiom. ∎

7 Subject reduction for β.

Proposition 29 β-reduction from β-expansion *The subtyping giving subject expansion for β in proposition 15 implies the subtyping giving the subject reduction for β, for singleton:*

$$\forall A\ \forall B \begin{pmatrix} \mathcal{S}_\tau(A(B)) \Rightarrow (\Lambda X\ A(X))[B] \neq \emptyset \Rightarrow \\ (A(B) \subset (\Lambda X\ A(X))[B]) \end{pmatrix}$$

Proof This is an immediate consequence of the definition of $\mathcal{S}_\subset(A(B))$, lemma 27 and proposition 15. ∎

The problem is to be able to apply the previous proposition ! To do so we need to be able to prove that types representing terms are singletons (if their variables are).

Proposition 30 Singleton and direct image *The direct image operator preserves the property of being a singleton:*

$$\forall A,B\ (\mathcal{S}_\tau(A) \Rightarrow \mathcal{S}_\tau(B) \Rightarrow \mathcal{S}_\tau(A[B]))$$

Proof We assume $\mathcal{S}_\tau(A)$ and $\mathcal{S}_\tau(B)$. First, we need to prove $A[B] \neq \emptyset$. This is easy using lemma 18. Then, we assume $A[B] \subset \bigcup X\ C(X)$ (i) and we need to prove $\exists X\ (A[B] \subset C(X))$. Using fact 11, from (i), we get $A \subset B \Rightarrow \bigcup X\ C(X)$, and using the commutation of singletons and $\mathcal{S}_\tau(B)$ we get $A \subset \bigcup Y\ (B \Rightarrow C(Y))$. Finally, using $\mathcal{S}_\tau(A)$, we find X such that $A \subset B \Rightarrow C(X)$, which gives $A[B] \subset C(X)$ using fact 11. ∎

Proposition 31 Singleton and abstraction *The abstraction operator preserves the property of being a singleton:*

$$\forall A \begin{pmatrix} \Lambda X\ A(X) \neq \emptyset \Rightarrow \forall X\ (\mathcal{S}_\tau(X) \Rightarrow \mathcal{S}_\tau(A(X))) \Rightarrow \\ \mathcal{S}_\tau(\Lambda X\ A(X)) \end{pmatrix}$$

Remark: the hypothesis $\Lambda X\ A(X) \neq \emptyset$ is necessary, because if $\Lambda X\ A(X)$ is not the representation of a λ-term, we could have $\forall X\ (X \neq \emptyset \Rightarrow A(X) \neq \emptyset)$ without having $\Lambda X\ A(X) \neq \emptyset$. However, we will always use this result when $\Lambda X\ A(X)$ is the representation of a λ-term. In this case, we will have $\vdash\ t\ :\ \Lambda X\ A(X)$ which implies $\Lambda X\ A(X) \neq \emptyset$.

Proof First, we prove $\forall A\ \forall B\ (A[\bigcup X\ (X \upharpoonright B(X))] \subset \bigcup X\ (A[X] \upharpoonright B(X)))$ using the property of image and union (fact 19) and the property of restriction (fact 9) and images (fact 11).

From this, using the atomicity and extensionality axioms, we deduce a stronger form of extensionality:

$$\forall A,B \begin{pmatrix} \mathcal{S}_\tau(B) \Rightarrow \forall X\ (\mathcal{S}_\tau(X) \Rightarrow (A[X] \subset B[X])) \Rightarrow \\ (A \subset B) \end{pmatrix} \text{(i)}.$$

Then, to prove the result, we assume $\Lambda X\ A(X) \neq \emptyset$, $\forall X\ (\mathcal{S}_\tau(X) \Rightarrow \mathcal{S}_\tau(A(X)))$ (ii), $\mathcal{S}_\tau(Y)$, $Y \subset \Lambda X\ A(X)$ and we must prove $(\Lambda X\ A(X) \subset Y)$. Using (i), we assume $Y \neq \emptyset$ and we prove $(\Lambda X_0\ A(X_0))[X] \subset Y[X]$. Using the subtyping corresponding to β-expansion, it is enough to prove $A(X) \subset Y[X]$. Then, using (ii), it is enough to prove $Y[X] \neq \emptyset$ and $Y[X] \subset A(X)$. The first one is immediate from $Y \subset \Lambda X_0\ A(X_0)$ and $Y \neq \emptyset$. The second one is a consequence of $\mathcal{S}_\tau(Y)$ and the subtyping corresponding to β-expansion which gives $(\Lambda X_0\ A(X_0))[X] \subset A(X)$. ∎

Fact 32 *From the left rules, we can derive the following rule:*

$$\frac{\Gamma, x : Y, \mathcal{S}_\tau(Y), Y \subset A \vdash t : B}{\Gamma, x : A \vdash t : B}\ \dagger$$

Proof Using classical reasoning to distinguish the case where P is true and the case where $\neg P$ is true, we easily derive the following rule:

$$\frac{\Gamma, x : Y, P \vdash t : B}{\Gamma, x : Y \upharpoonright P \vdash t : B}\ \dagger$$

Then, the wanted rule is a consequence of this, the atomicity axiom and the two left rules. ∎

Theorem 33 subject reduction for β *With all the axioms and rules, system ST enjoys the subject reduction for β*

Proof Let Γ be $x_1 : A_1, \ldots, x_n : A_n, P_1, \ldots, P_p$ a context. We assume $\Gamma \vdash t : B$ and t β-reducible to t'. We must prove $\Gamma \vdash t' : B$.

Let $Y_1, \ldots, Y_n$ be n new variables (not free in Γ or B). Let Γ' be $x_1 : Y_1, \ldots, x_n : Y_n, \mathcal{S}_\tau(Y_1), Y_1 \subset A_1, \ldots, \mathcal{S}_\tau(Y_n), Y_n \subset A_n, P_1, \ldots, P_p$. By the previous fact, it is enough to prove $\Gamma' \vdash t' : B$.

By theorem 14, this comes from $\Gamma' \vdash |t'|^\phi \subset B$ where $\phi(x_i) = Y_i$. From lemma 29 and fact 17, we easily get $|t'|^\phi \subset |t|^\phi$ (because we have enough hypotheses to show that for any subterm u of t or t', we have $\Gamma' \vdash \mathcal{S}_\tau(|u|^\phi)$ using propositions 30 and 31). Therefore, it is enough to prove $\Gamma' \vdash |t|^\phi \subset B$ which comes from theorem 14 and $\Gamma' \vdash t : B$. ∎

Theorem 34 subject expansion for η *With all the axioms and rules, system ST enjoys the subject expansion for η*

Proof The proof is similar to the proof of the previous theorem. We just need to prove

$$\forall A\ (\mathcal{S}_\tau(A) \Rightarrow (\Lambda X\ A[X] \subset A)),$$

which is an immediate consequence of the extensionality axiom and the subtyping giving subject expansion for β (fact 15).

Remark: $\forall A\ (\mathcal{S}_\tau(A) \Rightarrow (\Lambda X\ A[X] \subset A))$ is equivalent to the extensionality axiom and the hypothesis $\mathcal{S}_\tau(A)$ is necessary (otherwise, we can prove the incorrect version of the extensionality).

8 Formal realizability semantics

We will now prove a completeness theorem for our semantics. First, we need to formalize realizability, and we will use the same higher-order setting than for system ST.

The semantics we formalize here is a generalization of the semantics given in [16]. This semantics is parametrized by a relation R used to define realizability candidates. In [16] we used the $\beta\eta$-equivalence for R.

To prove our completeness theorem, we need a "formal" semantics. This means we will really define a formula in a specific system for "t realizes A". This kind of semantics was introduced by Krivine in [8] to be able to do induction on the proof of "t realizes A". This the key to our completeness proof: we will show that if we prove "$\vdash t$ realizes A" then we can prove $\vdash t : A$.

Definition 35 Formalization of realizability We consider a higher-order logic $\mathcal{M}$ with one sort o for propositions and one sort l for λ-terms.

We assume that there is a bijective mapping $t \mapsto \overline{t}$ between the λ-terms and the terms of sort l such that $x \mapsto \overline{x}$ is also a bijective mapping between λ-variables and the variables of sort l. Moreover, we assume that there is a constant @ of sort $l \to l \to l$ such that $@(\overline{t}, \overline{u}) = \overline{(t\,u)}$. We also want that x is free in t if and only if $\overline{x}$ is free in $\overline{t}$ and $\overline{t[x/u]} = \overline{t}[\overline{x}/\overline{u}]$.

This means that our logic $\mathcal{M}$ has a representation of λ-terms using objects of sort l.

However, it is important to remark that in $\mathcal{M}$, we have no way to identify λ-variables. This is important, because, as in Krivine's work [8], we use the logical variables of sort l in λ-terms as free variables. We may consider that, in fact, we use λ-terms with parameters and not really free variables. This implies that is we prove $\vdash A(x^l)$ in $\mathcal{M}$, then we can prove $\vdash \forall x^l A(x^l)$.

We assume that R is a constant symbol of sort $l \to l \to o$ and that there is a set of axioms T_R defining R.

To each sort s of system ST, we associate a sort $s^\star$ of $\mathcal{M}$ by replacing every occurrences of τ in s by $l \to o$.

For each sort s of system ST, we define ρ_s of sort $s^\star \to o$. $\rho_s(X)$ means that X is a valid interpretation for an object of sort s. We call such an object a **candidate**:

- $\rho_\tau := \lambda X^{l\to o} \forall x, y (X(x) \Rightarrow R(y,x) \Rightarrow X(y))$
- $\rho_o := \lambda X^o \forall Y^o (Y \Rightarrow Y)$ (always true)
- $\rho_{s\to s'} := \lambda F^{s^\star \to s'^\star} \forall X^{s^\star} (\rho_s(X) \Rightarrow \rho_{s'}(F(X)))$

To each expression e of sort s in system ST, we associate an expression $e^\star$ of sort $s^\star$ in $\mathcal{M}$ by induction as follows:

- $\Rightarrow^\star_\tau := \lambda a^{l\to o} \lambda b^{l\to o} \lambda x^l \forall y^l (a(y) \Rightarrow b(@(x,y)))$
- $\Rightarrow^\star_o := \Rightarrow$
- $\subset^\star := \lambda a^{l\to o} \lambda b^{l\to o} \forall x^l (a(x) \Rightarrow b(x))$
- $\Rrightarrow^\star := \lambda a^o \lambda b^{l\to o} \lambda x^l (a \Rightarrow b(x))$
- $\forall^{s\star}_\tau := \lambda a^{s^\star \to l\to o} \lambda x^l \forall z^{s^\star} (\rho_s(z) \Rightarrow a(z,x))$
- $\forall^{s\star}_o := \lambda a^{s^\star \to o} \forall z^{s^\star} (\rho_s(z) \Rightarrow a(z))$
- $(tu)^\star := (t^\star u^\star)$
- $(\lambda x^s\, t)^\star := \lambda x^{s^\star} t^\star$
- $x^{s\star}$ is given by a bijective mapping sending variables of sort s on variables of sort $s^\star$.

If A is a type of system ST and if t is a pure λ-term, we define $t \Vdash A := A^\star(t)$. This defines t **realizes** A.

If $W_1, \ldots, W_n$ are expressions or contexts of system ST, we define $\Delta(W_1, \ldots, W_n) = \rho_{s_1}(X_1^{s_1\star}), \ldots, \rho_{s_n}(X_n^{s_n\star})$ if $X_1^{s_1}, \ldots, X_n^{s_n}$ are the free variables of $W_1, \ldots, W_n$. $\Delta(W_1, \ldots, W_n)$ is a valid context of $\mathcal{M}$.

If $\Gamma = x_1 : A_1, \ldots, x_n : A_n, P_1, \ldots P_q$ is a valid context in system ST then, we define $\Gamma^\star = \overline{x_1} \Vdash A_1, \ldots, \overline{x_n} \Vdash A_n, P_1^\star, \ldots P_q^\star$. $\Gamma^\star$ is a valid context of $\mathcal{M}$.

Example 36 We can easily verify that $t \Vdash \forall X\ (X \Rightarrow X)$ means $\forall X^{l\to o}(\rho_\tau(X) \Rightarrow \forall x^l (X(x) \Rightarrow X(@(t,x))))$.

Remark on the formalization: there are many solutions to satisfy the conditions about objects of sort l: add function constants for all λ-terms and many axioms, or use a standard formalization of λ-calculus, for instance using De Bruijn indices (using indices only for bound variables and a lot of axioms), or use higher-order abstract syntax with a constant $\overline{\lambda}$ of sort $(l \to l) \to l$ to code λ. This is possible because the language of $\mathcal{M}$ is weak and there are not too much functions of sort $l \to l$. Moreover, this requires no special axioms. Note: combinatory logics is not suitable if you want to say precise things about R, but this is the simplest solution if R is the $\beta\eta$-equivalence.

We first prove the following lemma saying that if all variables in an expression A of system ST are interpreted by valid candidates then, A is a valid candidate (we need a condition on R):

Lemma 37 *Assume*

$$T_R \vdash \forall x^l \forall y^l \forall z^l (R(x,y) \to R(@(x,z), @(y,z)))$$

If A is an expression of sort s in system ST then, we get:

$$T_R, \Delta(A) \vdash \rho_s(A^\star)$$

Proof By induction on the size of the expression A, using the condition for the implication case. ∎

Lemma 38 *We prove in* $\mathcal{M} \vdash \forall X^{l\to o}(\rho_\tau(X) \Rightarrow (\mathcal{S}_\tau{}^\star(X) \Leftrightarrow \exists y^l \forall z^l (X(z) \Leftrightarrow R(z,y))))$.

Proof The right to left equivalence is easy. For the left to right equivalence, we instantiate the variable $B^\star$ in $\mathcal{S}_\tau{}^\star(X)$ with $\Psi = \lambda Y^{l\to o} \lambda x^l (Y(x) \wedge \exists y^l \forall z^l (Y(z) \Leftrightarrow R(z,y)))$.

Then, from the hypothesis $\rho_\tau(X)$, we prove $(X \subset \bigcup Y\, B(Y))^\star[B^\star = \Psi]$. Then, using the hypothesis $\mathcal{S}_\tau{}^\star(X)$ we find $Y^{l\to o}$ such that $(X \subset B(Y))^\star[B^\star = \Psi]$. X being non empty (from $\mathcal{S}_\tau{}^\star(X)$) we find y^l such that $\forall z^l (Y(z) \Leftrightarrow R(z,y))$ and $(X \subset Y)^\star$ which implies $\forall z^l (X(z) \Leftrightarrow R(z,y))$ using $\rho_\tau(X)$. ∎

Relation between ST and T_R: we associate to some axioms or rules of system ST a property of the relation R:

- The implication introduction rule for types: R contains the reduction of one redex, which means that for any λ-terms t and u we have:

$$T_R \vdash R(\overline{(\lambda x\, t)\, u}, \overline{t[x/u]})$$

- The $\forall$ elimination rules (for types and propositions):

$$T_R \vdash \forall x^l \forall y^l \forall z^l (R(x,y) \to R(@(x,z), @(y,z)))$$

- Commutation of singletons:

$$T_R \vdash \forall x^l \forall y^l \forall z^l (R(x,y) \to R(@(z,x), @(z,y)))$$

- Extensionality: the previous property and for any λ-terms t, u, v, with x not free in v,

$$T_R \vdash R(\overline{t}, \overline{u}) \text{ implies } T_R \vdash R(\overline{\lambda x\, t}, \overline{\lambda x\, u})$$

$$T_R \vdash R(\overline{v}, \overline{\lambda x (v\, x)}) \wedge R(\overline{\lambda x (v\, x)}, \overline{v})$$

- Complement: R is a symmetrical relation.

Theorem 39 Correctness of the semantics *Let ST' be a subsystem of ST and assume T_R allows to prove all the properties or R associated above to the axioms or rules of system ST'. Then, if we prove, in system ST', $\Gamma \vdash t : A$, we prove $\Gamma^\star, \Delta(\Gamma, A) \vdash \overline{t} \vDash A$ in $\mathcal{M}$ and if we prove, in system ST', $\Gamma \vdash P$, we prove $\Gamma^\star, \Delta(\Gamma, P) \vdash P^\star$ in $\mathcal{M}$.*

Proof By induction on the size of the proof. We give some details for a few axioms and rules:

- The $\forall$-elimination rules require lemma 37. It is not necessary for the other axioms because the translation of the quantifications bring the hypothesis that objects are candidates.

- The rules or axioms concerning $\bot_\tau$ or the union (except those mentioned below) are correct because $\bot_\tau$ is interpreted by the empty predicate and the union is really interpreted by a union.

- By lemma 38, if $x \vDash A$ then, $\lambda y(R(y,x))$ is a singleton and a subset of A, using $\rho_\tau(A^\star)$ which is given by the translation of the axiom.

- For the singleton commutation, we assume that X is a singleton and $B^\star$ a candidate for the sort $\tau \to \tau$. By lemma 38, there is x such that $y \vDash X$ equivalent to $R(y,x)$. We assume $z \vDash X \Rightarrow \bigcup Y\, B(Y)$ this implies that we find Y such that $@(z,x) \vDash B(Y)$. Then, by the property associated to the singleton commutation, we get $@(z,y) \vDash B(Y)$ for any y in X which implies $z \vDash X \Rightarrow B(Y)$ which gives the wanted result.

- For the extensionality axiom: we take two candidates $A^\star$ and $B^\star$ and assume that $\forall X\ (A[X] \subset B[X])^\star$, $\mathcal{S}_\tau(B)^\star$ and $x \vDash A$. By lemma 38, there is b such that $z \vDash X$ equivalent to $R(z,b)$. For $y \neq x$ and b, we define Y such that $u \vDash Y$ equivalent to $R(z,y)$. We have $@(x,y) \vDash A[Y]$ which implies $@(x,y) \vDash B[Y]$. This implies that $R(@(x,y), @(u,v))$ with $u \vDash B$ and $v \vDash Y$ that is $R(u,b)$ and $R(v,y)$. Let us write $\lambda y\, @(x,y)$ for $\overline{\lambda y'\, (\overline{x'\, y'})}$ with $x = \overline{x'}$ and $y = \overline{y'}$ which implies $\overline{(x'\, y')} = @(x,y)$. We use the same notation for $\lambda y\, @(u,v)$. Then, using the properties associated to this axiom, we have $R(x, \lambda y\, @(x,y))$, $R(\lambda y\, @(x,y), \lambda y\, @(u,v))$ and $R(\lambda y\, @(u,v), b)$, which implies that $x \vDash B$ (because $B^\star$ is a candidate). Remark: we have implicitly

y not free in b because y was chosen after b. We do not have to say that, because this is ensured by the formal constraint in the quantification rule of $\mathcal{M}$

- Complement axiom: we prove that $x \vDash A^c$ means that there is no y such that $R(x,y)$ and $y \vDash A$. Therefore, it is clear that if R is symmetrical then, $x \vDash A^c$ means that $x \nvDash A$ (because $A^\star$ is a candidate).

Theorem 40 *The singleton commutation, extensionality and complement axioms are all independent. That is they can not be proved using the two others and the other axioms and rules of system ST.*

Proof If T_R defines R to be the union of β-reduction and η-equivalence then, the complement axiom is false, but not the two others.

If T_R defines the β-equivalence then, the extensionality axiom is false, but not the two others.

If T_R defines the smallest equivalence relation containing the head-reduction and the η-equivalence then, the singleton commutation is false, but not the two others.

In the three cases above, it is easy to find the candidates showing that the concerned axiom is false looking at the proof of the correctness theorem. ∎

9 Completeness of the system ST.

We will prove completeness of system ST when R is the $\beta\eta$-equivalence. For all this section, we consider that T_R defines R to be the $\beta\eta$-equivalence.

Definition 41 Moving back to system ST To each sort s of $\mathcal{M}$, we associate a sort $s^\diamond$ of system ST by replacing every occurrences of l in s by τ.

To each sort s of $\mathcal{M}$, we associate an expression ν_s of sort $s^\diamond \to o$ by induction as follows:

- $\nu_l := \mathcal{S}_\tau$
- $\nu_o := \lambda x\, \top_o$
- $\nu_{s\to s'} := \lambda x^{s^\diamond \to s'^\diamond} \forall y^{s^\diamond}\ (\nu_s(y) \Rightarrow_\tau \nu_{s'}(x\, y))$

$\nu_s(x)$ means that x is a "correct" translation of an object of sort s of $\mathcal{M}$. This explains the translation of the quantification bellow.

To each expression e of sort s in $\mathcal{M}$, we associate an expression $e^\diamond$ of sort $s^\diamond$ in system ST by induction as follows:

- $\Rightarrow^\diamond := \Rightarrow_o$
- $\forall^{s^\diamond} := \lambda a^{s^\diamond \to o} \forall^{s^\diamond} z(\nu_s(z) \Rightarrow_o a(z))$
- $(tu)^\diamond := (t^\diamond u^\diamond)$
- $(\lambda x^s\, t)^\diamond := \lambda x^{s^\diamond}\, t^\diamond$
- $x^{s^\diamond}$ is given by a bijective mapping sending variables of sort s on variables of sort $s^\diamond$.
- $\overline{t}^\diamond := |t|^\phi$ with $\phi(x) = \overline{x}^\diamond$.
- $@^\diamond := \lambda X^\tau \lambda Y^\tau X[Y]$.
- $R^\diamond := \subset$.

If $W_1, \ldots, W_n$ are formulae or contexts of $\mathcal{M}$, we define $\Omega(W_1, \ldots, W_n) = \nu_s(x_1^{s_1^\diamond}), \ldots, \nu_s(x_p^{s_p^\diamond})$ where $x_1^{s_1}, \ldots, x_p^{s_p}$ are the free variables of $W_1, \ldots, W_n$. $\Omega(W_1, \ldots, W_n)$ is a valid context of system ST.

If $\Gamma = P_1, \ldots, P_q$ is a valid sequent in system $\mathcal{M}$ then, $\Gamma^\diamond := P_1^\diamond, \ldots, P_q^\diamond$. $\Gamma^\diamond$ is a valid context of system ST.

Lemma 42 *If we prove in $\mathcal{M}$, $T_R, \Gamma \vdash P$ then, we prove $\Gamma^\diamond, \Omega(\Gamma, P) \vdash P^\diamond$ in system ST.*

Proof First, we show that for any expression A of sort s in system $\mathcal{M}$, we can prove $\Omega(A) \vdash \nu_s(A^\diamond)$ in system ST (this is an easy induction on the size of A, using fact 30 and 31).

Using this, the proof is immediate by induction on the proof of $\Gamma \vdash P$, using the subtyping corresponding to $\beta\eta$-equivalence when using axioms in T_R. ∎

Definition 43 To each sort s of system ST, we associate an expression μ_s of sort $s \to s^{\star\diamond} \to o$ by induction as follows:

- $\mu_\tau := \lambda X^\tau \lambda Y^{\tau\to o} \forall Z\ (\mathcal{S}_\tau(Z) \Rightarrow ((Z \subset X) \Leftrightarrow Y(Z)))$
- $\mu_o := \lambda X^o \lambda Y^o (X \Leftrightarrow Y)$
- $\mu_{s\to s'} := \lambda X^{s\to s'} \lambda Y^{s^{\star\diamond}\to s'^{\star\diamond}}$
  $\forall x^s \forall y^{s^{\star\diamond}}\ (\mu_s(x,y) \Rightarrow_o \mu_{s'}(X(x), Y(y)))$

$\mu_s(X,Y)$ expresses that Y is equivalent to $X^{\star\diamond}$.

Lemma 44 *Let e be an expression of sort s in system ST. Let $x_1^{s_1}, \ldots, x_n^{s_n}$ be the free variables of e. Let Γ be $\mu_{s_1}(x_1^{s_1}, x_1^{s_1{}^{\star\diamond}}), \ldots, \mu_{s_n}(x_n^{s_n}, x_n^{s_n{}^{\star\diamond}})$. We can prove the following in system ST: $\Gamma \vdash \mu_s(e, e^{\star\diamond})$.*

Proof By induction on the size of e. All cases are consequences of the definition, we will treat only the most important case of $\Rightarrow_\tau$:

We assume $\mu_\tau(A,B)$, $\mu_\tau(A',B')$ and we must prove $\mu_\tau(A \Rightarrow A', B \Rightarrow^{\star\diamond} B')$. This means we assume $\mathcal{S}_\tau(X)$ and must prove $((X \subset A \Rightarrow A') \Leftrightarrow (B \Rightarrow^{\star\diamond} B')(X))$. Recall that we have

$$(B \Rightarrow^{\star\diamond} B')(X) = \forall Y\ (\mathcal{S}_\tau(Y) \Rightarrow B(Y) \Rightarrow B'(X[Y])).$$

- For the left to right implication, we assume $\mathcal{S}_\tau(Y)$, $B(Y)$ and prove $B'(X[Y])$ which is easy using facts 30, 11 and the hypotheses.
- For the left to right implication, we assume $(B \Rightarrow^{\star\diamond} B')(X)$ and prove $X \subset A \Rightarrow A'$. Using the atomicity axiom and fact 11, it is enough to prove $X[Y] \subset A'$ for any Y such that $\mathcal{S}_\tau(Y)$ and $Y \subset A$. This is easy using the hypotheses and fact 30. ∎

Definition 45 For any sort s of system ST, we define δ_s of sort $s \to s^{\star\diamond}$:

- $\delta_{s_1\to\ldots\to s_n\to\tau} := \lambda X^{s_1\to\ldots\to s_n\to\tau}\lambda y_1^{s_1^{\star\diamond}}\ldots\lambda y_n^{s_n^{\star\diamond}}\lambda z^\tau$
  $\forall x_1^{s_1}\ldots\forall x_n^{s_n}(\mu_{s_1}(x_1,y_1)\Rightarrow\ldots\Rightarrow\mu_{s_n}(x_n,y_n)$
  $\Rightarrow z\subset X(x_1,\ldots,x_n))$
- $\delta_{s_1\to\ldots\to s_n\to o} := \lambda X^{s_1\to\ldots\to s_n\to o}\lambda y_1^{s_1^{\star\diamond}}\ldots\lambda y_n^{s_n^{\star\diamond}}$
  $\forall x_1^{s_1}\ldots\forall x_n^{s_n}(\mu_{s_1}(x_1,y_1)\Rightarrow\ldots\Rightarrow\mu_{s_n}(x_n,y_n)$
  $\Rightarrow X(x_1,\ldots,x_n))$

Lemma 46 *For each sort s, we can prove in system ST:*

$$\vdash \forall x^s(\rho_s^\diamond(\delta_s(x))\wedge\nu_{s^\star}(\delta_s(x))\wedge\mu_s(x,\delta_s(x))$$

Proof Easy consequences of all the definitions. Remark: for ν_s, $\nu_{s^\star}(A)$ is always true because the sort $s^\star$ ends by o.∎

Theorem 47 *System ST is complete.*

Proof Let $\Gamma = x_1:A_1,\ldots,x_n:A_n,P_1,\ldots,P_q$ be a context of system ST, A a formula of system ST and t a λ-term.

We assume that $\Gamma^*,\Delta(\Gamma,A)\vdash(\overline{t}\vDash A)$ is provable in $\mathcal{M}$ and we prove that $\Gamma\vdash t:A$ is provable in system ST.

By the lemma 42, we have $\Gamma^{*\diamond},\Delta(\Gamma,A)^\diamond,\Omega(\Gamma^\star,A^\star)\vdash(\overline{t}\vDash A)^\diamond$ (1).

Let σ be the substitution which replaces every variable $x^{\star\diamond}$ of sort $s^{\star\diamond}$ by $\delta_s(x)$. Let B' be $B^{\star\diamond}\sigma$ for any formula B of system ST. Using the substitution lemma, we can apply the substitution σ to (1).

Then, we make the following remarks:

1. $\Delta(\Gamma,A)^\diamond$ contains all the formulae of the shape $\rho_s^\diamond(x^{\star\diamond})$ when x is a free variable of Γ or A of sort s. $\rho_s^\diamond$ being closed, when we apply the substitution σ we get $\rho_s^\diamond(\delta_s(x))$ which is provable by lemma 46.

2. $\Omega(\Gamma^\star,A^\star)$ is composed of two parts: the formulae $\mathcal{S}_\tau(\overline{x_i}^\diamond)$ for $i\in\{1,\ldots,n\}$ and all the formulae $\nu_s(x^{\star\diamond})$ when x is a free variable of Γ or A of sort s. When we apply the substitution σ, formulae of the shape $\mathcal{S}_\tau(\overline{x_i}^\diamond)$ are unchanged (because $\overline{x_i}^\diamond$ is a variable of sort τ and therefore unchanged by σ which substitutes only variables of sort $s^{\star\diamond}$ ending by o). The formulae of the second shape become $\nu_s(\delta_s(x))$ and are provable by lemma 46.

3. By lemmas 44 and 46, we have $\vdash\ \mu_\tau(A_i,A_i')$ and $\vdash\ \mu_o(P_i,P_i')$ which means, by definition of μ_s, $\mathcal{S}_\tau(\overline{x_i}^\diamond)\vdash(\overline{x_i}^\diamond\subset A_i)\Leftrightarrow A_i'(\overline{x_i}^\diamond)$ and $\vdash P_i\Leftrightarrow P_i'$.

4. We have $(\overline{t}\vDash A)^\diamond = A^{\star\diamond}(\overline{t}^\diamond)$. The free variable of t are among the $x_i^\diamond$ and are unchanged by the substitution σ. Therefore, applying σ to $(\overline{t}\vDash A)^\diamond$ gives $A'(\overline{t}^\diamond)$ which is equal to $A'(|t|^\phi)$ with $\phi(x)=\overline{x}^\diamond$ (by definition of $u\mapsto u^\diamond$). Thus, by lemma 44 and 46 we have $\mu_\tau(A,A')$ which implies $\mathcal{S}_\tau(|t|^\phi)\vdash(|t|^\phi\subset A)\Leftrightarrow A'(|t|^\phi)$

5. $\mathcal{S}_\tau(|t|^\phi)$ is provable using the hypothesis $\mathcal{S}_\tau(\overline{x_1}^\diamond),\ldots,\mathcal{S}_\tau(\overline{x_n}^\diamond)$ included in $\Omega(\Gamma^\star,A^\star)$ (by proposition 30 and 31).

These five remarks give $P_1,\ldots,P_n,\mathcal{S}_\tau(\overline{x_1}^\diamond),\overline{x_1}^\diamond\subset A_1,\ldots\mathcal{S}_\tau(\overline{x_n}^\diamond),\overline{x_n}^\diamond\subset A_n\vdash|t|^\phi\subset A$. Finally, using the rule in fact 32 and theorem 14 we get $\Gamma\vdash t:A$. ∎

References

[1] David Aspinall. Subtyping with Singleton Types. In *Computer Science Logic, CSL'94*, volume 933 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 1995.

[2] Stefano Berardi. Pruning simply typed λ-terms. *Journal of Logic and Computation*, 6:663–681, 1996.

[3] Judicaël Courant. Strong Normalization with Singleton Types. In Stefan Van Bakel, editor, *Second Workshop on Intersection Types and Related Systems*, volume 70 of *LNCS*, Copenhaguen, Danemark, July 2002. Elsevier.

[4] Philippe Curmin. *Marquage des preuves et extraction de programmes*. PhD thesis, Équipe de logique Université Paris VII, 1999.

[5] Jean-Yves Girard. The system F of variable types: fi fteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[6] Robert Harper and Christopher Stone. Deciding Type Equivalence with Singleton Kinds. In *Symposium on Principles of Programming Languages*, pages 214–227, 2000.

[7] Martin Hofmann and Benjamin Pierce. Positive subtyping. Technical report, LFCS, 1994. report number 94-303.

[8] Jean-Louis Krivine. *Lambda-Calculus : Types and Models*. Computers and their applications. Ellis Horwood, 1993.

[9] John C. Mitchell. Polymorphic type inference and containment. *Information and Computation*, 76:211–249, 1988.

[10] Catherine Parent. Synthesizing proofs from programs in the Calculus of Inductive Constructions. In *Mathematics for Programs Constructions*, volume 947 of *Lecture Notes in Computer Science*. Springer Verlag, 1995.

[11] Michel Parigot. Recursive programming with proofs. *Theoretical Computer Science*, 94:335–356, 1992.

[12] Christine Paulin-Mohring. Extracting F_ω's programs from proofs in the Calculus of Constructions. In *Sixteenth Annual ACM Symposium on Principles of Programming Languages*, Austin, January 1989. ACM.

[13] Christine Paulin-Mohring and Benjamin Werner. Synthesis of ML programs in system Coq. *Journal of Symbolic Logic*, 15:607–640, 1993.

[14] Benjamin C. Pierce. Programming with Intersection Types, Union Types, and Polymorphism. Technical Report CMU-CS-91-106, CMU, 1991.

[15] François Pottier. Type inference in the presence of subtyping: from theory to practice. Technical report, INRIA, 1998. report number:3483.

[16] Christophe Raffalli. System ST, Toward A Type System for Extraction AND Proof of Programs. prépublication du LAMA numéro 01-12c, to appear in APAL and available from the web page of the author, 2001.

[17] Christophe Raffalli. The PhoX proof assistant version 0.8. software available on the Internet: http://www.lama.univ-savoie.fr/~raffalli/phox.html, 2002.

[18] Paula Severi and Nora Szasz. Studies of a Theory of Specifi cations with built-in Program Extraction. *Journal of Automated Reasoning*, 27(1), July 2001.