A Semantical Storage Operator Theorem For All Types

Christophe Raffalli

LAMA - Équipe de Logique, Université de Savoie
73376 Le Bourget du Lac
email: raffalli@logique.jussieu.fr

Abstract

Storage operators are λ-terms which simulate call-by-value in call-by-name for a given set of terms. Krivine's storage operator theorem shows that any term of type $\neg D \to \neg D^*$, where D^* is the Gödel translation of D, is a storage operator for the terms of type D when D is a *data-type* or a formula with only positive second order quantifiers. We prove that a new semantical version of Krivine's theorem is valid for every types. This also gives a simpler proof of Krivine's theorem using the properties of data-types.

Key words: λ-calculus. Types. $\mathbf{AF}_2$ type system. Storage operators. Gödel translations.

1 Introduction.

The notion of storage operator was introduced by Krivine in [3]. A storage operator for a set of terms D is a term T simulating call-by-value in head-reduction for all elements in D: for t in D, $(T\,\varphi\,t)$ head-reduces to $(\varphi\,t_0)$ where $t_0 \sim_\beta t$ only depends on the normal form of t (the actual definition is slightly more complex).

The storage operator theorem is valid for a type D, if any term of type $\neg D \to \neg D^*$ is a storage operator for the elements of D (D^* being the Gödel translation of D). In [4] Krivine proves the storage operator theorem for the type of Church integer in the $\mathbf{AF}_2$ type system [5,8,9]. This result has been

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

extended to any $\forall$-*positive* type D (a type with only positive $\forall$-quantifiers) by Krivine [6] (using a semantical proof) and Nour [10] (using a syntactic proof). This result was originally proved with a specific Gödel translation (translating each atomic formula by its double negation). The Gödel translation was simplified to use only one negation (this is possible if we use no predicate constant in the logic) and finally the result was proved for a large variety of translations by Nour in [11].

Krivine also extends the theorem to classical logic in [7] (in classical logic, storage operators enjoy the property of translating classical proofs to intuitionistic ones) and Nour in [12] extends it to Parigot system TTR [16].

However, Nour gives a counter-example to the storage operator theorem for a type with a negative quantifier. We show that a slightly weaker form of the theorem is valid for any type in the $\mathbf{AF}_2$ type system (although there might be no term of type $\neg D \to \neg D^*$, but in this case the theorem is trivial). Our theorem uses a new notion of *semantical storage operator* which is equivalent to Krivine's notion for *data-types*. This new theorem gives a sufficient condition to ensure the existence of semantical storage operators for an arbitrary type D. Our proof also gives a simpler proof of Krivine's theorem for *data-types*.

The proof techniques we use is a simplification of Krivine's techniques. Using only 2 variables we give a sufficient condition for storage operators in pure λ-calculus (while Nour gives a condition using an extension: the directed λ-calculus [2]) and then a semantical proof allows us to conclude. One of the main innovation in our proof is the use of an asymmetrical Gödel translation: the atomic sub-formulas of A bound by a negative quantifier are left unchanged in the Gödel translation of A. We use semantical properties of this translation to prove our theorem.

After presenting our notation, we present in section 3 the notion of semantical storage operator and show how it is affected when we use it for $\forall$-positive types or data-types. In the next section, we show our sufficient condition for semantical storage operators in pure λ-calculus. In the section 5, we define the asymmetrical Gödel translation and prove its semantical properties. In the last section we finally prove our main theorem and discuss further possible works.

2 Notation.

Notation for the pure λ-calculus [1]:

- $(t_0\, t_1\, \ldots\, t_n)$ denotes applications. We write $(t^n\, u)$ for $(t(t(\ldots(t\, u))))$ with n applications of t.

- $\lambda x\, t$ denotes an abstraction. We write $(\lambda x\, t\, u)$ for $((\lambda x\, t)\, u)$.
- We write Λ for the set of all λ-terms.
- We know that any term u is of the form $\lambda x_1 \ldots \lambda x_n (t\, u_1\, \ldots\, u_n)$ where t is a variable or a redex. We call t the *head* of the term u.
- $t[x_1 \Leftarrow t_1, \ldots, x_n \Leftarrow t_n]$ denotes the simultaneous substitutions of the variables x_i by the terms t_i. We write [] for the identity substitution. We write $\sigma\sigma'$ for the simultaneous composition of two substitutions σ and σ': if $\sigma = [x_1 \Leftarrow t_1, \ldots, x_n \Leftarrow t_n]$ and $\sigma' = [x'_1 \Leftarrow t'_1, \ldots, x'_p \Leftarrow t'_p]$ with for all i, j $x_i \neq x'_j$ then $\sigma\sigma' = [x_1 \Leftarrow t_1, \ldots, x_n \Leftarrow t_n, x'_1 \Leftarrow t'_1, \ldots, x'_p \Leftarrow t'_p]$.
- $t \sim_\beta u$ (resp. $t \sim_{\beta\eta} u$) denotes the β-equivalence (resp. $\beta\eta$) and we write $t \rhd_\beta u$ for the β-reduction.
- $t \succ u$ denotes the weak-head-reduction: the smallest transitive and reflexive relation such that $(\lambda x\, t_0\, t_1\, \ldots\, t_n) \succ (t_0[x \Leftarrow t_1]\, t_2\, \ldots\, t_n)$.
- $\mathcal{V}(t)$ denotes the set of free variables of t.
- If A and B are sets, A^B is the set of all mappings from B to A (this has nothing to do with λ-calculus).

Proposition 1 *We use the following properties of weak-head-reduction:*

(1) For all terms t, u, v, if $t \succ u$ then $(t\, v) \succ (u\, v)$
(2) For all terms $t, t', u_1, \ldots, u_n$ if $t \succ t'$ then $t[x_1 \Leftarrow u_1, \ldots, x_n \Leftarrow u_n] \succ t'[x_1 \Leftarrow u_1, \ldots, x_n \Leftarrow u_n]$.
(3) For all terms t, u, if $t \sim_\beta \lambda x\, u$ then there exists $u' \sim_\beta u$ such that $t \succ \lambda x\, u'$.
(4) For all terms $t, u_1, \ldots, u_n$, if $t \sim_\beta (x\, u_1\, \ldots\, u_n)$ then there exist $u'_1 \sim_\beta u_1$, $\ldots$, $u'_n \sim_\beta u_n$ such that $t \succ (x\, u'_1\, \ldots\, u'_n)$.

The syntax of the $\mathbf{AF}_2$ type system [5] is defined as follows (the rules are given in table 1):

- We choose an infinite set $\mathcal{V}_1$ of first order variables.
- For each $n \in \mathbb{N}$, we choose an infinite set $\mathcal{V}_2^n$ of n-ary predicate variables.
- For each $n \in \mathbb{N}$, we choose a set $\mathcal{S}^n$ of n-ary function symbols.
- The set $\mathcal{T}$ of first order terms is defined as the smallest set such that:
  · $x \in \mathcal{T}$ if $x \in \mathcal{V}_1$.
  · $f(t_1, \ldots, t_n) \in \mathcal{T}$ if $f \in \mathcal{S}^n, t_1, \ldots, t_n \in \mathcal{T}$
- The set $\mathcal{F}$ of formulas is defined as the smallest set such that:
  · $X(t_1, \ldots, t_n) \in \mathcal{F}$ if $X \in \mathcal{V}_2^n, t_1, \ldots, t_n \in \mathcal{T}$
  · $A \to B \in \mathcal{F}$ if $A, B \in \mathcal{F}$
  · $\forall x\, A \in \mathcal{F}$ if $A \in \mathcal{F}, x \in \mathcal{V}_1$ (x is bound in $\forall x\, A$)
  · $\forall X\, A \in \mathcal{F}$ if $A \in \mathcal{F}, X \in \mathcal{V}_2^n$ (X is bound in $\forall X\, A$)
  · The equality $u = t$ if defined using Leibniz equality: $\forall X (X u \to X t)$
- Convention for substitutions: we use the letter χ to denote a variable of any kind (first order or second order). We use the letter φ associated to χ to

Table 1
The rules of the $\mathbf{AF}_2$ type system

$$\frac{}{x : F, \Gamma \vdash x : F}\, Ax \qquad \frac{\Gamma \vdash t : F[x \Leftarrow u] \qquad \Gamma \vdash \_ : u = v}{\Gamma \vdash t : F[x \Leftarrow v]} =$$

$$\frac{x : F, \Gamma \vdash t : G}{\Gamma \vdash \lambda x\, t : F \to G} \to_i \qquad \frac{\Gamma \vdash t : F \to G \qquad \Gamma \vdash u : F}{\Gamma \vdash (t\, u) : G} \to_e$$

$$\frac{\Gamma \vdash t : F \qquad \chi \notin \Gamma}{\Gamma \vdash t : \forall \chi\, F} \forall_i \qquad \frac{\Gamma \vdash t : \forall \chi F}{\Gamma \vdash t : F\,[\chi \Leftarrow \varphi]} \forall_e$$

denote an expression substitutable to χ (φ is a first order term if χ is a first order variable, φ is a n-ary predicate (predicates are written $\lambda x_1 \ldots \lambda x_n\, A$) if χ is a n-ary second order variable). Using this notation, simultaneous substitutions of any order can be written $A[\chi_1 \Leftarrow \varphi_1, \ldots, \chi_n \Leftarrow \varphi_n]$.

- We will sometimes write $X(\overline{t})$ for $X(t_1, \ldots, t_n)$ or $\lambda \overline{x}\, A$ for $\lambda x_1 \ldots \lambda x_n\, A$.

Here are two useful notions:

Definition 2 ($\forall$-positive) *We define by induction the sets of formulas $\mathcal{F}^+$ ($\forall$-positive types) and $\mathcal{F}^-$ as the smallest sets such that (ϵ denotes $+$ or $-$ and $\overline{\epsilon}$ denotes the opposite sign):*

- $X(\overline{t}) \in \mathcal{F}^\epsilon$ *for* $\epsilon = +$ *or* $\epsilon = -$.
- $A \to B \in \mathcal{F}^\epsilon$ *if* $B \in \mathcal{F}^\epsilon$ *and* $A \in \mathcal{F}^{\overline{\epsilon}}$.
- $\forall x\, A \in \mathcal{F}^\epsilon$ *if* $A \in \mathcal{F}^\epsilon$.
- $\forall X\, A \in \mathcal{F}^+$ *if* $A \in \mathcal{F}^+$.

Definition 3 (2-closed) *We say that a formula is 2-closed if it has no free second order variable.*

The standard semantics of the $\mathbf{AF}_2$ type system is defined as follows:

- let $\mathcal{C}$ be a set of subsets of Λ. We say that $\mathcal{C}$ is a set of candidates if it is closed for implications (if $\Phi, \Phi \in \mathcal{C}$ then $\Phi \to \Phi' = \{t \mid \forall u \in \Phi, (t\, u) \in \Phi'\} \in \mathcal{C}$), infinite intersections and weak-head-expansion ($\Phi \in \mathcal{C}$, $u \in \Phi$ and $t \succ u$ implies $t \in \Phi$).
- When $\mathcal{C}$ is a set of candidates, a $\mathcal{C}$-interpretation $\mathcal{I}$ is defined by
  - A mapping from first order variables to terms, $x \mapsto |x|^{\mathcal{I}}_{\mathcal{C}}$,
  - A mapping from n-ary function symbols to Λ^{Λ^n}, written $f \mapsto |f|^{\mathcal{I}}_{\mathcal{C}}$.
  - A mapping from n-ary predicate variables to $\mathcal{C}^{\Lambda^n}$, written $X \mapsto |X|^{\mathcal{I}}_{\mathcal{C}}$.
- If χ is any kind of variable and if ψ is a possible interpretation for χ then for any $\mathcal{C}$-interpretation $\mathcal{I}$, we write $\mathcal{I}[\chi \Leftarrow \psi]$ for the $\mathcal{C}$-interpretation defined

by $|\chi|_{\mathcal{C}}^{\mathcal{I}[\chi \Leftarrow \psi]} = \psi$ and $|\chi'|_{\mathcal{C}}^{\mathcal{I}[\chi \Leftarrow \psi]} = |\chi'|_{\mathcal{C}}^{\mathcal{I}}$ if $\chi' \neq \chi$.

- $\mathcal{C}$-interpretations are extended to every term and formula by:
  - $|f(t_1, \ldots, t_n)|_{\mathcal{C}}^{\mathcal{I}} = |f|_{\mathcal{C}}^{\mathcal{I}}(|t_1|_{\mathcal{C}}^{\mathcal{I}}, \ldots, |t_n|_{\mathcal{C}}^{\mathcal{I}})$
  - $|X(t_1, \ldots, t_n)|_{\mathcal{C}}^{\mathcal{I}} = |X|_{\mathcal{C}}^{\mathcal{I}}(|t_1|_{\mathcal{C}}^{\mathcal{I}}, \ldots, |t_n|_{\mathcal{C}}^{\mathcal{I}})$
  - $|A \to B|_{\mathcal{C}}^{\mathcal{I}} = |A|_{\mathcal{C}}^{\mathcal{I}} \to |B|_{\mathcal{C}}^{\mathcal{I}} = \{u \in \Lambda \mid \forall v \in |A|_{\mathcal{C}}^{\mathcal{I}}, (u\ v) \in |B|_{\mathcal{C}}^{\mathcal{I}}\}$
  - $|\forall X\ A|_{\mathcal{C}}^{\mathcal{I}} = \bigcap_{\Phi \in \mathcal{C}^{\Lambda^n}} |A|_{\mathcal{C}}^{\mathcal{I}[X \Leftarrow \Phi]}, (X \in \mathcal{V}_2^n)$
  - $|\forall x\ A|_{\mathcal{C}}^{\mathcal{I}} = \bigcap_{t \in \Lambda} |A|_{\mathcal{C}}^{\mathcal{I}[x \Leftarrow t]}$

We will use the following definitions and results about the semantics:

Definition 4 *We define $\mathcal{C}_s$ as the set of all subsets of Λ closed for $\sim_\beta$. It is easy to see that $\mathcal{C}_s$ is a set of candidates. We call a* standard interpretation *any $\mathcal{C}_s$-interpretation and we omit $\mathcal{C}_s$ in the notation.*

Proposition 5 *For any $\mathcal{C}$-interpretation $\mathcal{I}$ and any formulas A, $|A|_{\mathcal{C}}^{\mathcal{I}} \in \mathcal{C}$.*

Proposition 6 *For any $\mathcal{C}$-interpretation $\mathcal{I}$ and any formulas, $|A[x \Leftarrow t]|_{\mathcal{C}}^{\mathcal{I}} = |A|_{\mathcal{C}}^{\mathcal{I}[x \Leftarrow |t|_{\mathcal{C}}^{\mathcal{I}}]}$ and $|A[X \Leftarrow \lambda \overline{x}\, B]|_{\mathcal{C}}^{\mathcal{I}} = |A|_{\mathcal{C}}^{\mathcal{I}[X \Leftarrow \lambda \overline{x} |B|_{\mathcal{C}}^{\mathcal{I}[\overline{x} \Leftarrow \overline{z}]}]}$.*

Proposition 7 (Adequation lemma) *If $x_1 : A_1, \ldots, x_n : A_n \vdash t : A$ then for all $\mathcal{C}$-interpretation $\mathcal{I}$ and all $u_1 \in |A_1|_{\mathcal{C}}^{\mathcal{I}}, \ldots, u_n \in |A_n|_{\mathcal{C}}^{\mathcal{I}}$ we have $t[x_1 \Leftarrow u_1, \ldots, x_n \Leftarrow u_n] \in |A|_{\mathcal{C}}^{\mathcal{I}}$.*

Definition 8 *A formula $D[x]$ with one free variable x is a data-type*[2] *for a standard interpretation $\mathcal{I}$ if for all term $t \in |D[u]|^{\mathcal{I}}$ we have $t \sim_\beta |u|^{\mathcal{I}}$ and $\exists v \in \mathcal{T}, \exists t' \sim_\beta t$ such that $\vdash t' : D[v]$.*

Proposition 9 *The formula $N[n] = \forall X(X(0) \to \forall y(X(y) \to X(Sy)) \to X(n))$ is a data-type for a standard interpretation $\mathcal{I}$ if and only if $|0|^{\mathcal{I}} \sim_\beta \lambda x \lambda f\, x$ and $|S|^{\mathcal{I}}(\overline{n}) \sim_\beta \overline{n+1}$ for all Church numeral $\overline{n} = \lambda x \lambda f(f^n x)$. Moreover, this proposition is proved using the following result: $t \in |N[u]|^{\mathcal{I}}$ implies $\exists n \in \mathbb{N}, t \sim_\beta \overline{n}$.*

The proofs of last three propositions can be found in [5] for the standard semantics. The general proofs are nearly identical and can be found in [17].

3 Storage operators and semantics.

Here are Krivine's notions of storage operators:

Definition 10 (Storage operator for a term or a set of terms) *A term T is a storage operator for a closed term t if there exists a term $u \sim_\beta t$, such*

[2] this definition is slightly stronger than the one in [5]

that for all $\varphi \in \Lambda$ and $v \sim_\beta t$ we have $(T\,\varphi\,v) \succ (\varphi\,(u\sigma))$ for a substitution σ.

A term T is a storage operator for a set of closed terms, if it is a storage operator for each element in the set.

Here is Nour's definition:

Definition 11 (Storage operator for a type) *A term T is a storage operator for a type A if for any term t such that $\vdash t : A$, there exists two terms u, u' with $u \sim_\beta u'$ and $\vdash u' : A$, such that for all $\varphi \in \Lambda$ and $v \sim_\beta t$ we have $(T\,\varphi\,v) \succ (\varphi\,(u\sigma))$ for a substitution σ.*

The second definition is weaker that the first one, but for data-types, they are equivalent: if T is a storage operator for all the types $N[t]$ then T is a storage operator for each Church numeral using the proposition 9. The second definition is needed because the first one is too restrictive in the general case due to its relation with the β-equivalence.

Nour shows that for any $\forall$-positive type A, if $\vdash T : \neg_O A \to \neg_O A^*$ (see the section 5) then T is a storage operator for the type A. However, for more general types, this theorem does not hold. For the type[3] $D = \forall X(\forall Y(Y \to X) \to X)$, the term

$$T = \lambda f \lambda x(x\ \lambda y \lambda f(f\ \lambda z(z\,y))\ f) : \neg_O D \to \neg_O D^*$$

is not a storage operator for D with the previous definition. Indeed, we have $t = \lambda x(x\,\theta) : D$ for any typable term θ and $(T\,f\,t) \succ (f\,\lambda z(z\,\theta))$. However, the term $\lambda z(z\,y)$ acts as $\lambda z(z\,\theta)$ in any *well behaving context* (because the sub-term θ will in fact never appear in head position). Therefore, with a more general definition, we can say that T is a storage operator for D. To give this definition, we need the following notion:

Definition 12 (extension) *If γ is a finite set of variables, we define by induction the (non symmetric) relation $\rightsquigarrow_\gamma$ the smallest relation such that:*

- *If $u = (x\,u_1\,\ldots\,u_n)$ with $x \notin \gamma$ then $u \rightsquigarrow_\gamma t$ for any term t.*
- *For all variables x, $x \rightsquigarrow_\gamma x$.*
- *If $u \rightsquigarrow_\gamma t$ and $u' \rightsquigarrow_\gamma t'$ then $(u\,u') \rightsquigarrow_\gamma (t\,t')$.*
- *If $u \rightsquigarrow_{\gamma\cup\{x\}} t$ then $\lambda x\,u \rightsquigarrow_\gamma \lambda x\,t$*
- *$u \rightsquigarrow t$ if and only if $u \rightsquigarrow_\emptyset t$ and we say that t is an extension of u.*

Proposition 13 *For all λ-terms t', t and any finite set of variables γ, if $t \rightsquigarrow_\gamma t'$ and $\mathcal{V}(t) \subset \gamma$ then $t' = t$.*

[3] due to K. Nour

Proof: By induction on the definition of $t \rightsquigarrow_\gamma t'$. ■

Proposition 14 *For all λ-terms t, t', u, u', any finite set of variables γ and any λ-variable x, $t \rightsquigarrow_{\gamma \cup x} t'$ and $u \rightsquigarrow_\gamma u'$ imply $t[x \Leftarrow u] \rightsquigarrow_\gamma t'[x \Leftarrow u']$.*

Proof: By induction on the definition of $t \rightsquigarrow_{\gamma \cup x} t'$. ■

Proposition 15 *For all λ-terms t, u, u' and any finite set of variables γ, if $u \rightsquigarrow_\gamma t$ and $u \rhd_\beta u'$ (resp. $u \succ u'$) then there exists t' such that $t \rhd_\beta t'$ (resp. $t \succ t'$) and $u' \rightsquigarrow_\gamma t'$.*

Proof: By induction on the length of the reduction, and for one step reduction by induction on the structure of the term u, using the proposition 14 in the redex case. ■

Now we can give the wanted definition:

Definition 16 (Semantical storage operator for a type) *A term T is a semantical storage operator for a type A if for all closed term t such that $\vdash t : A$ there exists two terms u, u' satisfying*

- *$u' \rightsquigarrow u$,*
- *for all standard interpretation $\mathcal{I}$, $u' \in |A|^{\mathcal{I}}$ and*
- *for all terms $\varphi \in \Lambda$ and $v \sim_\beta t$, $(T\,\varphi\,v) \succ (\varphi\,(u\sigma))$ for a substitution σ.*

Therefore, T is a semantical storage operator for a type A, if it can compute any t of type A into a canonical element which is an extension of a term semantically of the same type and which only depends on the β-normal form of t. But non β-equivalent terms of the same type can be stored on different canonical forms.

Our main theorem 38 says that for any type A, any term T of type $\neg_O A \to \neg_O A^+$ (B^+ denotes a particular Gödel translation of B) is a semantical storage operator for the type A. Thus, a term T of type $\forall \overline{x}(\neg_O A[\overline{x}] \to \neg_O A^+[\overline{x}])$ is a semantical storage operator for $A[t_1, \ldots, t_n]$ for any $t_1, \ldots, t_n$.

A question arise: why do we need the notion of extension ? As shown by the proposition 18, we can always choose $u' = u$ if A is a $\forall$-positive type. This corollary is also true for other types (like D given above). But we do not know if it is true in general. We believe it to be true, but we think that a counter-example to proposition 18 for a formula satisfying the hypothesis of the theorem 38 would show that the notion of extension is necessary.

To justify the name of storage operator, we must show that the substitution is not used in any computation. For $\forall$-positive types this is easy because we

can prove the following:

Proposition 17 *For any standard interpretation $\mathcal{I}$ and any 2-closed $\forall$-positive type A, if $t \in |A|^{\mathcal{I}}$ then t is β-equivalent to a closed term.*

Proof: We assume $t \in |A|^{\mathcal{I}}$ for a standard interpretation $\mathcal{I}$. Let us choose $x_1, \ldots, x_n, \ldots$ infinitely many variables not free in t. We define the set of candidates $\mathcal{C}_\chi$ by $\Phi \in \mathcal{C}_\chi$ if and only if:

(1) Φ is closed under $\beta\eta$-equivalence.
(2) $\Phi \subset \mathcal{X}$ where $\mathcal{X}$ is defined by $t \in \mathcal{X}$ if and only if t is β-equivalent to a term whose free variables ares among $x_1, \ldots, x_n, \ldots$
(3) For all $\Phi \in \mathcal{C}_\chi, x_i \in \mathcal{X}$ and $t_1, \ldots, t_p \in \mathcal{X}$, $(x_i\, t_1\, \ldots\, t_p) \in \Phi$.

Then it is easy to see that an intersection of a family of elements in $\mathcal{C}_\chi$ is in $\mathcal{C}_\chi$. Moreover, if $\Phi, \Phi' \in \mathcal{C}_\chi$ then $\Phi \to \Phi' \in \mathcal{C}_\chi$: (1) is immediate. For (2), we choose t in $\Phi \to \Phi'$, by (3) we have $x_i \in \Phi$ with $x_i \in \mathcal{X}$ and x_i not free in t, so we have $(t\, x_i) \in \Phi' \subset \mathcal{X}$ which implies $t \sim_\eta \lambda x_i (t\, x_i) \in \mathcal{X}$. For (3) we have, for all $x_i \in \mathcal{X}$, $t_1, \ldots, t_p \in \mathcal{X}$ and $u \in \Phi$, $(x_i\, t_1\, \ldots\, t_p\, u) \in \Phi'$ (using (2) for Φ and (3) for Φ'). Thus $\mathcal{C}_\chi$ is a set of candidates which implies $|A|^{\mathcal{I}}_{\mathcal{C}_\chi} \subset \mathcal{X}$ using proposition 5.

Moreover, we can easily show by induction on B, that if $B \in \mathcal{F}^+$ ($\forall$-positive type) (resp. $B \in \mathcal{F}^-$), if $\mathcal{C} \subset \mathcal{C}'$ are two sets of candidates and if $\mathcal{I}'$ is a $\mathcal{C}$-interpretation (and therefore a $\mathcal{C}'$-interpretation) then we have $|B|^{\mathcal{I}'}_{\mathcal{C}'} \subset |B|^{\mathcal{I}'}_{\mathcal{C}}$ (resp. $|B|^{\mathcal{I}'}_{\mathcal{C}} \subset |B|^{\mathcal{I}'}_{\mathcal{C}'}$). This is easy because of the position of the second order quantifiers and because $\mathcal{C} \subset \mathcal{C}'$ implies that an intersection on $\mathcal{C}'$ in a subset of an intersection on $\mathcal{C}$.

Then we can apply this to $\mathcal{C}_\chi \subset \mathcal{C}_s$ and for any $\mathcal{C}_\chi$-interpretation $\mathcal{I}'$ coinciding with $\mathcal{I}$ on first order term, we find $|A|^{\mathcal{I}} = |A|^{\mathcal{I}'} \subset |A|^{\mathcal{I}'}_{\mathcal{C}_\chi} \subset \mathcal{X}$ (because A is 2-closed). This implies $t \in \mathcal{X}$ and therefore t is β-equivalent to a closed term (because $x_1, \ldots, x_n, \ldots$ were chosen not free in t). ■

Therefore if u is β-equivalent to a closed term the computation of $(\varphi\, u)$ and $(\varphi\, u\sigma)$ are identical and the substitution is never used (because the free variables of u can never appear in head position).

In the general case we need to use a less formal argument to deduce the same thing: in any *well behaving context* a computation never brings a free variable in head position. We mean by well behaving context, for instance, a context which computes to a data-type or to $\lambda x\, x$ producing some *output* using *side-effects*[4].

[4] we can introduce side-effects by extending the λ-calculus with constants c_s such that $(c_s\, t_1\, \ldots\, t_n) \succ ((\lambda x\, x)\, t_1\, \ldots\, t_n)$ producing the *output* s

For instance, if $\vdash \varphi : A \to N[t]$, then $(\varphi\, u\sigma)$, $(\varphi\, u)$ and $(\varphi\, u')$ will weak-head-reduce in the same way if $u' \rightsquigarrow u$ and $u' \in |A|^{\mathcal{I}}$ for any interpretation $\mathcal{I}$. Indeed, using the propositions 9 and 15, we find that $(\varphi\, u') \succ t'$ implies there exists $t' \rightsquigarrow t$ such that $(\varphi\, u) \succ t$ and therefore, $(\varphi\, u\sigma) \succ t\sigma$. Therefore, σ will never be used during the weak-head-reduction because $(\varphi\, u')$ never reduces to a term whose head is a free variable.

However in arbitrary contexts, we can not justify the name of storage operator and the type $D = \forall X(\forall Y(Y \to X) \to X)$ given before gives a counter-example.

Proposition 18 *For any standard interpretation $\mathcal{I}$ and any 2-closed $\forall$-positive type A, $t \in |A|^{\mathcal{I}}$ and $t \rightsquigarrow t'$ imply $t' \in |A|^{\mathcal{I}}$.*

Proof: We choose a standard interpretation $\mathcal{I}$ and a 2-closed $\forall$-positive type A. Let t, t' be terms such that $t \in |A|^{\mathcal{I}}$ and $t \rightsquigarrow t'$. Using the proposition 17 and the Church-Rosser theorem, we find a closed term t'' such that $t \triangleright_\beta t''$ and $t'' \in |A|^{\mathcal{I}}$. Thus by the propositions 15 and 13 we have $t' \triangleright_\beta t''$ which implies $t' \in |A|^{\mathcal{I}}$. ■

Corollary 19 *For any $\forall$-positive data-type $D[x]$, if for every first-order term u, T is a semantical storage operator for $D[u]$ then T is storage operator for the set of elements of D.*

Proof: Easy using the definition 8 and the proposition 18. ■

4 bl-reduction and storage operators.

In this section, to prove the theorem 38, we will show a characterization of storage operators in pure λ-calculus. To do this we choose two λ-variables b and l. We use them to define a particular reduction (the bl-reduction). To know if a term T is a storage operator for a term t, we will only need to perform a simple test on the bl-reduction of T applied to a variable f and a special transformed form of t (using many occurrences of b and l).

Definition 20 (bl-simplification: $\searrow$) *To define the bl-simplification we define by induction a family of relations $\searrow^\alpha$ where α is a finite set of λ-variables:*

- *If x is a variable, $x \neq b, l$ then $x \searrow^\alpha x$.*
- *If $t \searrow^\alpha u$ and $t' \searrow^\alpha u'$ then $(t\, t') \searrow^\alpha (u\, u')$.*
- *If $t \searrow^{\alpha\cup\{x\}} u$ then $\lambda x\, t \searrow^\alpha \lambda x\, u$.*
- *If $t \searrow^\alpha u$ and $\mathcal{V}(\lambda x\, t) \cap \alpha = \emptyset$ then $\lambda x\, t \searrow^\alpha (l\, \lambda x\, u)$.*
- *If $t \searrow^\alpha u$, $t' \succ t$ and $\mathcal{V}(t') \cap \alpha = \emptyset$ then $t' \searrow^\alpha (b\, u)$.*

- $t \searrow u$ *if and only if* $t \searrow^{\emptyset} u$.

The intuitive meaning of $t \searrow u$, is that u is obtained from t by β-reduction, but all sub-terms where weak-head-reductions were needed are marked by a variable b in u. Moreover, if an abstraction binds a variable occurring inside a sub-term where reductions were needed, it is marked by a l. This restriction on the abstraction and the variable l is only used to prove the lemma 28. If we simplify the previous definition to use only the variable b, with no restriction for the abstraction case, then we can still get a weaker version of the theorem 38.

It is useful (but we do not use it directly) to remark that $t' \searrow^{\alpha} t$ implies $b, l \notin \mathcal{V}(t')$ and $\mathcal{V}(t) \subset \mathcal{V}(t') \cup \{b, l\}$.

Lemma 21 *If* $\mathcal{V}(t') \cap \beta = \emptyset$ *then* $t' \searrow^{\alpha} t$ *if and only if* $t' \searrow^{\alpha \cup \beta} t$.

Proof: The proof is an immediate induction, using the fact that if $t \succ u$ then $\mathcal{V}(u) \subset \mathcal{V}(t)$. ■

Lemma 22 *For all terms* t, u, t', u' *and all variable* $x \neq b, l$, *if* $t' \searrow t$ *and* $u' \searrow u$ *then* $t'[x \Leftarrow u'] \searrow t[x \Leftarrow u]$.

Proof: This follows from $t' \searrow^{\alpha} t$ and $u' \searrow u$ with $\mathcal{V}(u') \cap \alpha = \emptyset$ implies $t'[x \Leftarrow u'] \searrow^{\alpha} t[x \Leftarrow u]$. We prove this by induction on the definition of $t' \searrow^{\alpha} t$:

- If $t = t' = y$, $y \in \mathcal{V}_0$ ($y = x$ or $y \neq x$) the result is immediate using the hypothesis $\mathcal{V}(u') \cap \alpha = \emptyset$ and the lemma 21.
- If $t' = (v'\, w')$ and $t = (v\, w)$ with $v' \searrow^{\alpha} v$ and $w' \searrow^{\alpha} w$ then using the induction hypothesis we get $v'[x \Leftarrow u'] \searrow^{\alpha} v[x \Leftarrow u]$ and $w'[x \Leftarrow u'] \searrow^{\alpha} w[x \Leftarrow u]$ and we get the expected result.
- If $t' = \lambda y\, v'$ and $t = \lambda y\, v$ with $v' \searrow^{\alpha \cup \{y\}} v$ then we must choose $y \neq x$ not free in u', thus by induction hypothesis we have $v'[x \Leftarrow u'] \searrow^{\alpha \cup \{y\}} v[x \Leftarrow u]$. This gives the wanted result.
- If $t' = \lambda x\, v'$, $t = (l\, \lambda x\, v)$, $v' \searrow^{\alpha} v$ and $\mathcal{V}(t') \cap \alpha = \emptyset$ then we have by induction hypothesis $v'[x \Leftarrow u'] \searrow^{\alpha} v[x \Leftarrow u]$. Moreover, $\mathcal{V}(t'[x \Leftarrow u']) \subset \mathcal{V}(t') \cup \mathcal{V}(u')$ implies $\mathcal{V}(t'[x \Leftarrow u']) \cap \alpha = \emptyset$. Thus we get the expected result.
- If $t' \succ v'$, $t = (b\, v)$, $v' \searrow^{\alpha} v$ and $\mathcal{V}(t') \cap \alpha = \emptyset$ then we have by induction hypothesis $v'[x \Leftarrow u'] \searrow^{\alpha} v[x \Leftarrow u]$. Moreover, we have $t'[x \Leftarrow u'] \succ v'[x \Leftarrow u']$, which gives the wanted result using $\mathcal{V}(t'[x \Leftarrow u']) \cap \alpha = \emptyset$ as in the previous case. ■

Definition 23 (bl-reduction: $\succ_b$) *The bl-reduction is the smallest transitive and reflexive relation* $\succ_b$ *such that:*

- *for all terms* u v, $u \succ v$ *implies* $u \succ_b v$

- *for all terms* $u_0, \ldots, u_n$ *(*$n \geq 0$*),* $(b\, u_0 \ldots u_n) \succ_b (u_0 \ldots u_n)$
- *for all terms* $u_0, \ldots, u_n$ *(*$n \geq 1$*) and variable* x*,* $(l\ (\lambda x\, u_0)\ u_1 \ldots u_n) \succ_b (u_0[x \Leftarrow u_1]\ u_2 \ldots u_n)$

Lemma 24 *For any terms* t, t', u *if* $t' \searrow t$ *and* $t \succ_b u$ *then there exists a term* u' *such that* $u' \searrow u$ *and* $t' \succ u'$*.*

Proof: We prove this result by induction on the bl-reduction $t \succ_b u$:

- If $t = (b\, u_0 \ldots u_n)$ and $u = (u_0 \ldots u_n)$ then by definition of $t' \searrow t$, $t' = (u'_0\, u'_1 \ldots u'_n)$ with $u'_0 \succ u''_0$, $u''_0 \searrow u_0$ and for all $1 \leq i \leq n$, $u'_i \searrow u_i$. Thus we can choose $u' = (u''_0\, u'_1 \ldots u'_n)$ and we have the wanted result.
- If $t = (l\, (\lambda x\, u_0)\, u_1 \ldots u_n)$ and $u = (u_0[x \Leftarrow u_1] \ldots u_n)$ then by definition of $t' \searrow t$, $t' = ((\lambda x\, u'_0)\, u'_1 \ldots u'_n)$ with for all $0 \leq i \leq n$, $u'_i \searrow u_i$. Thus we can choose $u' = (u'_0[x \Leftarrow u'_1] \ldots u'_n)$ and we have the wanted result using lemma 22.
- If $t = ((\lambda x\, u_0)\, u_1 \ldots u_n)$ and $u = (u_0[x \Leftarrow u_1] \ldots u_n)$ then by definition of $t' \searrow t$, $t' = ((\lambda x\, u'_0)\, u'_1 \ldots u'_n)$ with $u'_0 \searrow^{\{x\}} u_0$ and for all $1 \leq i \leq n$, $u'_i \searrow u_i$. Then using the proof of lemma 22 (more precisely the result we proved by induction to establish this lemma) we find $u'_0[x \Leftarrow u'_1] \searrow^{\{x\}} u_0[x \Leftarrow u'_1]$ (because we can choose x not free in u'_1). So we have x not free in $u'_0[x \Leftarrow u'_1]$ implies $u'_0[x \Leftarrow u'_1] \searrow u_0[x \Leftarrow u'_1]$ using lemma 21. Thus we can choose $u' = (u'_0[x \Leftarrow u'_1] \ldots u'_n)$.
- Both cases for the reflexivity and transitivity are immediate. ■

Definition 25 (bl-saturation) *For any normal term* t *with* b *and* l *not free in* t*, we define its bl-saturation* t^* *by induction as follows:*

- *If* $t = (x\, t_1 \ldots t_n)$ *then* $t^* = (b\, (x\, t^*_1 \ldots t^*_n))$.
- *If* $t = (\lambda x\, u)$ *then* $t^* = (b\, (l\, \lambda x\, u^*))$.

Lemma 26 *For all terms* t, u *with* t *normal and* b*,*l *not free in* t *and* u*, we have:* $u \triangleright_\beta t$ *implies* $u \searrow t^*$ *(The converse is also true, but we do not need it).*

Proof: We prove this by induction on the term t. We are in one of the following cases:

- If $t = (x\, t_1 \ldots t_n)$ then $u \triangleright_\beta t$ implies $u \succ (x\, u_1 \ldots u_n)$ with, for all i, $u_i \triangleright_\beta t_i$. Thus by induction hypothesis we get, for all i, $u_i \searrow t^*_i$. Therefore, $u \searrow (b\, (x\, t^*_1 \ldots t^*_n)) = t^*$.
- If $t = (\lambda x\, t')$ then $u \triangleright_\beta t$ implies $u \succ (\lambda x\, u')$ with $u' \triangleright_\beta t'$. Thus by induction hypothesis we get $u' \searrow t'^*$ and $u \searrow (b\, (l\, \lambda x\, t'^*)) = t^*$ follows. ■

Corollary 27 *For all terms* t, u*, if* t *is normal,* $u \sim_\beta t$ *implies* $u \searrow t^*$*.*

Proof: Immediate using the lemma 26 and the Church-Rosser theorem. ■

Lemma 28 *For any term v, there exists a term v_0 with $v \rightsquigarrow v_0$ such that for all term $u \searrow v$ we have $u = v_0\sigma$ for a substitution σ.*

Proof: The idea of the proof is to replace every maximum sub-term of v of the form $(b\,t)$ or $(l\,t)$ by a fresh variable. We deduce the lemma from the following result: for any term v and all disjoint sets of variable γ,β with $b, l \notin \gamma \cup \beta$, β infinite and $\beta \cap \mathcal{V}(v) = \emptyset$, there exists a term v_0 such that:

(1) $v \rightsquigarrow_\gamma v_0$
(2) $\mathcal{V}(v_0) - \mathcal{V}(v) \subset \beta$
(3) for all u, $u \searrow^\gamma v$ implies $u = v_0\sigma$ for a substitution σ with $x\sigma = x$ if $x \notin \beta \cap \mathcal{V}(v_0)$ and $\mathcal{V}(x\sigma) \cap \gamma = \emptyset$ if $x\sigma \neq x$.

We prove this by induction on the term v:

- If $v = (b\,t)$ or $v = (l\,t)$, we choose $v_0 = x$ for $x \in \beta$ (thus (1) and (2) are verified). Then for any term $u \searrow^\gamma v$ we have $u = v_0[x \Leftarrow u]$. By definition of $u \searrow^\gamma v$ we have $\mathcal{V}(u) \cap \gamma = \emptyset$ which gives (2).
- If $v = x$, we choose $v_0 = x$ and we get the wanted result (remark: if $x = b$ or $x = l$ the hypothesis $u \searrow^\gamma v$ is never satisfied).
- If $v = (v'\,v'')$ with $v' \neq b, l$, we apply the induction hypothesis on v' with γ, β to find v'_0 and we apply it again on v'' with $\gamma, \beta - \mathcal{V}(v'_0)$ to find v''_0 and we choose $v_0 = (v'_0\,v''_0)$. By definition of $v \rightsquigarrow_\gamma v_0$, we have (1). Then we have (2) from the induction hypothesis.

  Let us choose $u \searrow^\gamma v$, by definition we have $u = (u'\,u'')$ with $u' \searrow^\gamma v'$ and $u'' \searrow^\gamma v''$. By induction hypothesis, we find σ', σ'' with $u' = v'_0\sigma'$, $u'' = v''_0\sigma''$, $x\sigma' = x$ if $x \notin \beta \cap \mathcal{V}(v'_0)$ and $x\sigma'' = x$ if $x \notin (\beta - \mathcal{V}(v'_0)) \cap \mathcal{V}(v''_0)$. Therefore if $x\sigma' \neq x$ then $x \in \mathcal{V}(v'_0)$ and thus $x\sigma'' = x$ and if $x\sigma'' \neq x$ then $x \notin \mathcal{V}(v'_0)$ and thus $x\sigma' = x$. We can define $\sigma = \sigma'\sigma''$ the simultaneous composition of σ' and σ'' (see the notation section). Moreover, if $x \in \mathcal{V}(v'_0)$ then we have $x\sigma'' = x$ (because $x \notin (\beta - \mathcal{V}(v'_0)) \cap \mathcal{V}(v''_0)$) and if $x \in \mathcal{V}(v''_0)$ then we have $x\sigma' = x$ (either $x \in \mathcal{V}(v'') \subset \mathcal{V}(v)$ which implies $x \notin \beta$ or $x \notin \mathcal{V}(v'')$ which implies $x \in \beta - \mathcal{V}(v'_0)$ and therefore $x \notin \beta \cap \mathcal{V}(v'_0)$). Thus we have $v_0\sigma = (v'_0\sigma'\sigma''\,v''_0\sigma'\sigma'') = (v'_0\sigma'\,v''_0\sigma'') = (u'\,u'') = u$ and $x\sigma = x$ if $x \notin \beta \cap \mathcal{V}(v'_0)$.

  We have now to check that if $x\sigma \neq x$ then $\mathcal{V}(x\sigma) \cap \gamma = \emptyset$. This is immediate using the induction hypothesis because $x\sigma = x\sigma'$ or $x\sigma = x\sigma''$.
- If $v = \lambda y\,v'$ we apply the induction hypothesis on v' with $\gamma \cup \{y\}$,β (we must choose $y \neq b, l$) to find a term v'_0. Let us choose $u \searrow^\gamma v$, by definition we have $u = (\lambda y\,u')$ with $u' \searrow^{\gamma\cup\{x\}} v'$. We can find σ such that $u' = v'_0\sigma$, $x\sigma = x$ if $x \notin \beta \cap \mathcal{V}(v_0)$ and $\mathcal{V}(x\sigma) \cap (\gamma \cup \{y\}) = \emptyset$ if $x\sigma \neq x$. Therefore, if we choose $v_0 = \lambda y\,v'_0$, we have $u = \lambda y\,(v'_0\sigma) = (\lambda y\,v'_0)\sigma$ and σ satisfies the expected properties. ■

Definition 29 ($\succ_b$-storage operator) *A term T is a $\succ_b$-storage for a type A if for all closed normal term t such that $\vdash t : A$ there exists a term u such that $u \in |A|^{\mathcal{I}}$ for any standard interpretation $\mathcal{I}$ and $(T\ f\ t^*) \succ_b (f\ u)$ where f is a variable not free in T and t.*

Theorem 30 *If T is a $\succ_b$-storage operator for a type A then it is a semantical storage operator for A.*

Proof: Let us assume that T is a $\succ_b$-storage operator for A, we choose t such that $\vdash t : A$. Then, if t_0 is the normal form of t, we also have $\vdash t_0 : A$ (using the normalization and subject reduction for the $\mathbf{AF}_2$ type system [5]) and we can find u' such that $u' \in |A|^{\mathcal{I}}$ for any standard interpretation $\mathcal{I}$ and $(T\ f\ t_0^*) \succ_b (f\ u')$. Now using the lemma 28 we can find u such that $u' \rightsquigarrow u$ and for all term $w \searrow u'$ there exists a substitution σ with $w = u\sigma$. Let us choose an arbitrary term $t' \sim_\beta t \sim_\beta t_0$. This implies that $t' \searrow t_0^*$ (by lemma 27). Therefore, we have $(T\ f\ t') \searrow (T\ f\ t_0^*)$. Then using lemma 24, we find t'' such that $(T\ f\ t') \succ t''$, $t'' \searrow (f\ u')$. This implies that $t'' = (f\ w)$ with $w \searrow u'$ and therefore $w = u\sigma$ for a substitution σ. ■

5 Gödel translations.

In this section and the following, we choose a particular variable O used to define some Gödel translations. We will write $\neg_O A$ for $A \to O$.

Definition 31 (Gödel translation) *The Gödel translation A^* of a formula A is defined by induction as follows:*

- $X(\overline{t})^* = \neg_O X(\overline{t})$
- $(A \to B)^* = A^* \to B^*$
- $(\forall x\ A)^* = \forall x\ A^*$
- $(\forall X\ A)^* = \forall X\ A^*$ *(if $X \neq O$)*

Definition 32 (Asymmetric Gödel translation) *We define by induction the mappings $A \mapsto A^+$, and $A \mapsto A^-$. In what follows, ϵ stands for $+$ or $-$ and $\overline{\epsilon}$ denotes the opposite of the sign ϵ:*

- $X(\overline{t})^\epsilon = X(\overline{t})$
- $(A \to B)^\epsilon = A^{\overline{\epsilon}} \to B^\epsilon$.
- $(\forall x\ A)^\epsilon = \forall x\ A^\epsilon$.
- $(\forall X\ A)^+ = \forall X\ A^+[X \Leftarrow \lambda x_1 \ldots x_n\ \neg_O X(x_1, \ldots, x_n)]$ *(if X is n-ary).*
- $(\forall X\ A)^- = \forall X\ A^-$.

The idea of the asymmetric Gödel translation $A \mapsto A^+$ is to transform atomic formulas of the form $X(\overline{t})$ by $X(\overline{t}) \to O$ when X is bound by a positive quantifier and to let it unchanged if it is free or bound by a negative quantifier. We need to use the mapping $A \mapsto A^-$ to give a proper inductive definition.

Proposition 33 *For any 2-closed $\forall$-positive type A, $A^* = A^+$.*

Proof: We prove by simultaneous induction that for all formulas $A \in \mathcal{F}^+$ (resp. $A \in \mathcal{F}^-$) if $X_1, \ldots, X_n$ are the free second order variables of A then $A^* = A^+[X_1 \Leftarrow \lambda\overline{x}\, \neg_O X_1(\overline{x}), \ldots, X_n \Leftarrow \lambda\overline{x}\, \neg_O X_n(\overline{x})]$ (resp. $A^* = A^-[X_1 \Leftarrow \lambda\overline{x}\, \neg_O X_1(\overline{x}), \ldots, X_n \Leftarrow \lambda\overline{x}\, \neg_O X_n(\overline{x})]$). This induction is immediate and we get the expected result if A is 2-closed. ■

Definition 34 *We define $\mathcal{C}_O$ as the set of all $\Phi \subset \Lambda$ closed for β-equivalence and bl-expansion ($\Phi \in \mathcal{C}_O, t \in \Phi, t' \succ_b t$ imply $t' \in \Phi$).*

Proposition 35 *$\mathcal{C}_O$ is a set of candidates. Moreover, $\Phi \to \Phi'$ is closed for bl-expansion if Φ' is (we need no hypothesis on Φ).*

Proof: First, it is immediate that the intersection of a family of subsets of Λ closed for β-equivalence and bl-expansion is itself closed for the same relations. Then Φ closed for β-equivalence or bl-expansion implies Φ closed for β-expansion. Now we must prove that for all $\Phi \subset \Lambda$ and $\Phi' \in \mathcal{C}_O$ we have $\Phi \to \Phi' \in \mathcal{C}_O$. Let us choose $t \in \Phi \to \Phi'$ and t' such that $t' \succ_b t$ (resp. $t' \sim_\beta t$). We must show $t' \in \Phi \to \Phi'$. We choose $u \in \Phi$ and we have $(t\, u) \in \Phi'$ and $(t'\, u) \succ_b (t\, u)$ (resp. $(t'\, u) \sim_\beta (t\, u)$) which imply $(t'\, u) \in \Phi'$. ■

Proposition 36 *For any formula A and any $\mathcal{C}_O$-interpretation $\mathcal{I}$ (thus $\mathcal{I}$ is also a standard interpretation because $\mathcal{C}_O \subset \mathcal{C}_s$) we have*

$$|A^-|^{\mathcal{I}} \subset |A|^{\mathcal{I}}_{\mathcal{C}_O} \subset |A^+|^{\mathcal{I}}$$

Proof: We prove this result by induction on the formula A. The atomic case and the first order quantifier cases are immediate. For the implication, we have $A = B \to C$ and the induction hypothesis give $|B^-|^{\mathcal{I}} \subset |B|^{\mathcal{I}}_{\mathcal{C}_O} \subset |B^+|^{\mathcal{I}}$ and $|C^-|^{\mathcal{I}} \subset |C|^{\mathcal{I}}_{\mathcal{C}_O} \subset |C^+|^{\mathcal{I}}$. Then we get the expected result because $A^+ = B^- \to C^+$ and $A^- = B^+ \to C^-$. For the second order quantification case, we have $A = \forall X\, B$ with X of arity n and, by induction hypothesis, for any $\Phi \in \mathcal{C}_O^{\Lambda^n}$, $|B^-|^{\mathcal{I}[X \Leftarrow \Phi]} \subset |B|^{\mathcal{I}[X \Leftarrow \Phi]}_{\mathcal{C}_O} \subset |B^+|^{\mathcal{I}[X \Leftarrow \Phi]}$. From this, we get the expected result because $\mathcal{C}_O^{\Lambda^n} \subset \mathcal{C}_s^{\Lambda^n}$ implies

$$|A^-|^{\mathcal{I}} = \bigcap_{\Phi \in \mathcal{C}_s^{\Lambda^n}} |B^-|^{\mathcal{I}[X \Leftarrow \Phi]} \subset \bigcap_{\Phi \in \mathcal{C}_O^{\Lambda^n}} |B^-|^{\mathcal{I}[X \Leftarrow \Phi]}$$

and because $\{\lambda\overline{x}(\Phi(\overline{x}) \to |O|^{\mathcal{I}})|\Phi \in \mathcal{C}_s^{\Lambda^n}\} \subset \mathcal{C}_O^{\Lambda^n}$ (consequence of proposition 35 and $|O|^{\mathcal{I}} \in \mathcal{C}_O$) and $|B^+[X \Leftarrow \lambda\overline{x}\,\neg_O X(\overline{x})]|^{\mathcal{I}[X\Leftarrow\Phi]} = |B^+|^{\mathcal{I}[X\Leftarrow\lambda\overline{x}(\Phi(\overline{x})\to|O|^{\mathcal{I}})]}$ (by lemma 6) imply

$$\bigcap_{\Phi\in\mathcal{C}_O^{\Lambda^n}} |B^+|^{\mathcal{I}[X\Leftarrow\Phi]} \subset \bigcap_{\Phi\in\mathcal{C}_s^{\Lambda^n}} |B^+|^{\mathcal{I}[X\Leftarrow\lambda\overline{x}(\Phi(\overline{x})\to|O|^{\mathcal{I}})]} = |A^+|^{\mathcal{I}}$$

■

Lemma 37 *If t is a normal term and if A is a 2-closed formula such that $\vdash t : A$ then for any standard interpretation $\mathcal{I}$ if $|O|^{\mathcal{I}}$ is closed under bl-expansion (and β-equivalence) then we have $t^* \in |A^+|^{\mathcal{I}}$.*

Proof: We will say that a substitution σ and a $\mathcal{C}_O$-interpretation $\mathcal{I}$ satisfy a context $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, where b and l do not appear, if for all i we have $x_i\sigma \in |A_i|^{\mathcal{I}}_{\mathcal{C}_O}$, $b\sigma = b$ and $l\sigma = l$.

We first prove by induction that if $\Gamma \vdash t : A$ then for any $\sigma, \mathcal{I}$ satisfying Γ we have $t^*\sigma \in |A|^{\mathcal{I}}_{\mathcal{C}_O}$. The proof is very similar to the proof of the adequation lemma 7. All the cases except the arrow introduction are identical (we can add a b whenever we want using propositions 35 and 5). For the arrow introduction case, we have $A = B \to C$, $t = \lambda x\, v$ and $\Gamma, x : B \vdash v : C$. We choose $\sigma, \mathcal{I}$ satisfying Γ and u such that $u \in |B|^{\mathcal{I}}_{\mathcal{C}_O}$. We can always assume $x \neq b, l$, $x\sigma = x$ and x not free in $y\sigma$ if $y \neq x$ (otherwise we change x in the all proof). Thus we have $(v^*\sigma)[x \Leftarrow u] = v^*(\sigma[x \Leftarrow u])$ ($\sigma[x \Leftarrow u]$ is the simultaneous composition of σ and $[x \Leftarrow u]$, see the notation section). Then by induction hypothesis, we find $v^*(\sigma[x \Leftarrow u]) \in |C|^{\mathcal{I}}_{\mathcal{C}_O}$. Thus we have $(b\,(l\ \lambda x\, v^*\sigma)\, u) \succ_b (v^*)(\sigma[x \Leftarrow u])$ implies $(b\,(l\ \lambda x\, v^*\sigma)\, u) \in |C|^{\mathcal{I}}_{\mathcal{C}_O}$ using lemmas 35 and 5. Therefore, as u was arbitrary, we have $t^*\sigma = (b\,(l\ \lambda x\, v^*\sigma)) \in |A|^{\mathcal{I}}_{\mathcal{C}_O}$.

To finish the proof, let t be a normal term and A be a 2-closed formula such that $\vdash t : A$. Then for any standard interpretation $\mathcal{I}$ we can choose $\mathcal{C}_O$-interpretation $\mathcal{I}'$ coinciding with $\mathcal{I}$ on O and on first order terms and we have $t^* \in |A|^{\mathcal{I}'}_{\mathcal{C}_O}$. Finally, the lemma 36 gives $t \in |A^+|^{\mathcal{I}'} = |A^+|^{\mathcal{I}}$ (because O is the only free variable of A^+). ■

6 The storage operator theorem.

Theorem 38 *For any 2-closed formula A, if $\vdash T : \neg_O A \to \neg_O A^+$ then T is a semantical storage operator for A.*

Proof: Using the proposition 30 we just need to show that T is a $\succ_b$ storage operator for A. Let t be a closed normal term such that $\vdash t : A$. Then we

choose a variable f not free in T and any standard interpretation $\mathcal{I}$ such that

$$t \in |O|^{\mathcal{I}} \text{ if and only if there exists } v \in |A|^{\mathcal{I}} \text{ with } t \succ_b (f\ v).$$

Then by definition $|O|^{\mathcal{I}}$ is closed under bl-expansion and it is easy to show that it is also closed for β-equivalence (because $\succ_b$ contains $\succ$ and because $(f\ v)$ is in weak-head-normal form). Using lemma 37 we have $t^* \in |A^+|^{\mathcal{I}}$. Moreover, we have $f \in |A \to O|^{\mathcal{I}}$. Therefore, by hypothesis on T we have $(T\ f\ t^*) \in |O|^{\mathcal{I}}$. Thus there exists v such that $(T\ f\ t^*) \succ_b (f\ v)$ and $v \in |A|^{\mathcal{I}}$. Moreover, as the bl-reduction is deterministic, v is independent on $\mathcal{I}$. Thus for any standard interpretation $\mathcal{I}'$ we have $v \in |A|^{\mathcal{I}'}$ (because O is not free in A). ■

Further outlook The theorem 38 gives a sufficient condition for a term T to be a semantical storage operator for A. Nevertheless, there are a few questions that arise from this result:

- Is the notion of extension necessary (this is discussed in the section 3) ?
- Our theorem uses a particular form of Gödel translation. We could generalize it to any usual translation [15] as long as we leave atomic formulas bound by a negative second order quantifier unchanged (this does not matter for $\forall$-positive types !). However is the theorem still true for symmetric Gödel translation ?
- If $\vdash T : A \to A^+$ then T can store any term typable of type A. But is T a storage operator for any term semantically of type A or do there exist a type A and a term $t \in |A|^{\mathcal{I}}$ for all standard interpretations $\mathcal{I}$ such that T does not work for t ? A recent work by Nour and Farkh [14] shows the completeness of the standard semantics for a large subset of the $\forall$-positive types, which solves the problem for these formulas.
- Another question can be answered in the same way: for which types are the notions of semantical storage operator for a type A and storage operator for a type A equivalent?

References

[1] H. P. Barendregt. *The Lambda Calculus: Its Syntax and Semantics.* North-Holland, revised edition, 1984.

[2] K. Nour & R. David. Storage operator and directed λ-calculus. *Journal of Symbolic Logic*, 60(4):1050–1086, 1995.

[3] J.-L. Krivine. Opérateurs de mise en mémoire et traduction de Godël. Technical Report 3, Equipe de Logique de Paris 7, December 1989.

[4] J.-L. Krivine. Opérateurs de mise en mémoire et traduction de gödel. *Archive for Mathematical Logïc*, 30:241–267, 1990.

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

[6] J.-L. Krivine. Mise en mémoire, preuve général. Manuscript, 1993.

[7] J.-L. Krivine. Classical logic, storage operators and second order λ-calculus. *APAL*, 1994.

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

[9] D. Leivant. Reasoning about functional programs and complexity classes associated with type disciplines. In *24th Annual Symposium on Foundations of Computer Science*, volume 44, pages 460–469, 1983.

[10] K. Nour. *Opérateurs de mise en mémoire en lambda-calcul pur et typé.* PhD thesis, Université de Savoie, 1993.

[11] K. Nour. A general type for storage operator. *Mathematical Logic Quaterly*, 41:505–514, 1995.

[12] K. Nour. Storage operator and $\forall$-positive types in system ttr. *Mathematical Logic Quaterly*, 42, 1996. number 2.

[13] K. Nour. Storage operators and $\forall$-positive types. *Informatique Théorique et Application*, 30(3):261–293, 1996.

[14] S. Farkh & K. Nour. Résultats de complétude pour des classes de types du système af2. Submitted to Theoretical Informatics and Applications, 1997.

[15] M. Parigot. Strong normalization for second order classical natural deduction. In *Logic in Computer Sciences*, pages 39–46, 1993.

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

[17] Christophe Raffalli. *L' Arithmétiques Fonctionnelle du Second Ordre avec Points Fixes.* PhD thesis, Université Paris 7, Février 1994.