Electronic Notes in Theoretical Computer Science 70 No. 1 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume70.html 19 pages

Strong Normalization with Singleton Types

Judicaël Courant [1]

LRI, CNRS UMR 8623
Bât 490, Université Paris Sud
F-91405 Orsay Cedex

Abstract

This paper presents a new lambda-calculus with singleton types, called $\lambda^{\beta\delta}_{\leq\{\}}$. The main novelty of $\lambda^{\beta\delta}_{\leq\{\}}$ is the introduction of a new reduction, the δ-reduction, replacing any variable declared of singleton type by its value, and the definition of equality as the syntactic equality of $\beta\delta$-normal forms. The δ-reduction has a very odd behavior on untyped terms, which renders its metatheoretical study difficult since the usual proof method for subject-reduction and Church-Rosser property are inapplicable. Nevertheless, these properties can be proved simultaneously with strong normalization on typed terms using a proof method *à la* Coquand-Gallier, borrowing ideas to Goguen. In spite of its complex metatheory, our calculus enjoys a simple, sound and complete type-inference algorithm.

1 Introduction

A singleton type $\{M\}_A$ is the subtype of A whose elements are equal to M for some notion of equality. Type systems with singleton types help giving a theoretical account of ML-like module systems and their compilation [8], or of definitions in type theory [1].

In [1], Aspinall raises the difficult question of the decidability of type-checking in the presence of singletons; he remarks that this question merely reduces to the question of testing equality of terms in his system. He suggests that one could first define a reduction replacing any variable declared of type $\{M\}_A$ by M, and study it together with β-reduction to show that two given terms are equal if and only if they have equal normal forms.

This paper presents a variant of Aspinall's $\lambda_{\leq\{\}}$, called $\lambda^{\beta\delta}_{\leq\{\}}$, in which we define and investigate this reduction and the decidability of type-checking.

Section 2 introduces $\lambda^{\beta\delta}_{\leq\{\}}$. Section 3 demonstrates the odd behavior of the reduction on untyped terms and sketches the metatheory of $\lambda^{\beta\delta}_{\leq\{\}}$. Section 4

[1] Email: Judicael.Courant@lri.fr

presents sound and complete type inference and type checking algorithms for $\lambda^{\beta\delta}_{\leq\{\}}$. Finally, related works are discussed in Section 5.

2 Definition of $\lambda^{\beta\delta}_{\leq\{\}}$

In this section, the $\lambda^{\beta\delta}_{\leq\{\}}$-calculus is defined. The $\lambda^{\beta\delta}_{\leq\{\}}$-calculus results from the addition of singletons to the simply-typed lambda calculus.

2.1 Syntax

Figure 1 presents the syntax of $\lambda^{\beta\delta}_{\leq\{\}}$. $\lambda^{\beta\delta}_{\leq\{\}}$ has a singleton type $\{M\}_A$ denoting the type of elements of type A that are convertible to M. Moreover, the arrow of the simply typed lambda-calculus is replaced by a dependent product.

Types			
A	$::=$	P	The Atomic Type
	$\mid$	$\{M\}_A$	Singletons
	$\mid$	$\Pi x\!:\!A.A$	Products
Terms			
M	$::=$	x	Variables
	$\mid$	$\lambda x\!:\!A.M$	Abstractions
	$\mid$	$(M\ M)$	Applications
Contexts			
Γ	$::=$	ϵ	
	$\mid$	$\Gamma; x : A$	

where x ranges over a set of variables.

Fig. 1. Grammar of $\lambda^{\beta\delta}_{\leq\{\}}$

This syntax is the same as Aspinall's $\lambda_{\leq\{\}}$, excepted that $\lambda^{\beta\delta}_{\leq\{\}}$ has only one atomic type P whereas Aspinall considers P ranges over a set of primitive types. This difference is irrelevant for our purpose.

We do not want to deal with name capture issues. Therefore, terms are always assumed in Barendregt convention and the variables declared in a given context are distinct.

2.2 Reductions

We define here $\lambda^{\beta\delta}_{\leq\{\}}$ (untyped) reductions.

The intended meaning of singleton types is the following: each time a variable x is declared with a type $\{M\}_A$, the convertibility relation is extended with the equality $x = M$.

We choose to describe the equalities generated by singleton declarations with a reduction relation, called δ-reduction. This reduction is parameterized by a context. We note $\Gamma \vdash M\ \delta\ M'$ to mean that M is a δ-redex reducing to M' in the context Γ and $\Gamma \vdash M \rhd_\delta M'$ to mean that M δ-reduces to M'.

How should we define this reduction? As we already said, we have $\Gamma \vdash x\ \delta M$ if $x : \{M\}_A$ appears in Γ. But we also want more: for instance, if $x : \Pi x' : A_1.\{M\}_{A_2}$ appears in Γ, we would like $(x\ M')$ to δ-reduce to $M\{x' \leftarrow M'\}$ since it belongs to the type $(\{M\}_{A_2})\{x' \leftarrow M'\}$.

One could imagine defining the δ-reduction as the relation such that $\Gamma \vdash M\ \delta\ M'$ whenever M has (principal) type $\{M'\}_A$ in Γ, but we do not adopt this approach since it makes typing and reduction mutually recursive. Indeed typing requires some term comparisons.

Instead, we introduce a judgment, $\Gamma \vdash_{pp} M : A$, read "in the context Γ, M has pre-principal type A", with the rules given Figure 2. We define M to be a δ-redex reducing to M' in Γ (noted $\Gamma \vdash M\ \delta\ M'$) if $\Gamma \vdash_{pp} M : \{M'\}_A$ is derivable for some A using the rules given Figure 2.

$$\mathsf{D/VAR}\frac{\Gamma(x) = A}{\Gamma \vdash_{pp} x : A}$$

$$\mathsf{D/APP}\frac{\Gamma \vdash_{pp} M_1 : \Pi x{:}A_1.A_2}{\Gamma \vdash_{pp} (M_1\ M_2) : A_2\{x \leftarrow M_2\}}$$

Fig. 2. pre-principal type inference

Notice the rule D/APP does not check anything about M_2. Indeed, checking M_2 has type A_1 would introduce a mutual dependency between typing and reduction. On the opposite, the rules for pre-principal type inference do not rely on term comparison, are syntax-directed, and give each term M at most one type A.

It is clear that one can decide whether there exists A such that $\Gamma \vdash_{pp} M : A$ and can even compute this unique A if it exists, as the rules are syntax-directed and for each of them the subject of its premise is a strict subterm of the subject of its conclusion. We can now formally define the δ-reduction as follows:

- δ-reduction in one step of M to M' in a context Γ is noted $\Gamma \vdash M \rhd_\delta M'$ and is defined as the least monotonic relation including δ. As the δ-reduction depends on a context, monotonicity has to be understood as follows:
  - if $\Gamma; x : A \vdash M_1 \rhd_\delta M_2$, then $\Gamma \vdash \lambda x{:}A.M_1 \rhd_\delta \lambda x{:}A.M_2$;
  - if $\Gamma; x : A \vdash A_1 \rhd_\delta A_2$, then $\Gamma \vdash \Pi x{:}A.A_1 \rhd_\delta \Pi x{:}A.A_2$;
  - if $\Gamma \vdash A_1 \rhd_\delta A_2$, then $\Gamma \vdash \lambda x{:}A_1.M \rhd_\delta \lambda x{:}A_2.M$, $\Gamma \vdash \Pi x{:}A_1.A \rhd_\delta \Pi x{:}A_2.A$, and $\Gamma \vdash \{M\}_{A_1} \rhd_\delta \{M\}_{A_2}$;
  - if $\Gamma \vdash M_1 \rhd_\delta M_2$, then $\Gamma \vdash (M_1\ M) \rhd_\delta (M_2\ M)$, $\Gamma \vdash (M\ M_1) \rhd_\delta (M\ M_2)$, and $\Gamma \vdash \{M_1\}_A \rhd_\delta \{M_2\}_A$.
- β-reduction is defined as usual.
- $\beta\delta$-reduction in one step in a context Γ is the union of β-reduction and

δ-reduction in Γ and is noted $\triangleright_{\beta\delta}$. Its reflexive transitive closure is noted $\triangleright^{\star}_{\beta\delta}$.

- the convertibility relation in a context Γ is noted $\Gamma \vdash M_1 \bowtie M_2$ and is defined as

$$\exists M\ \Gamma \vdash M_1 \triangleright^{\star}_{\beta\delta} M \wedge \Gamma \vdash M_2 \triangleright^{\star}_{\beta\delta} M$$

We do not define convertibility as the least congruence containing $\triangleright_{\beta\delta}$ as this congruence is the total relation, as shown in section 3.1.2.

We can now give examples of δ-reduction. Let us define $\Gamma = x_0 : P; x_1 : \{x_0\}_P; x_2 : \Pi y\!:\!P.\{y\}_P; x_3 : P$.

- In Γ, x_1 is a δ-redex reducing to x_0 since $\Gamma \vdash_{pp} x_1 : \{x_0\}_P$.
- In Γ, $(x_2\ x_3)$ is a δ-redex reducing to $y\{y \leftarrow x_3\} = x_3$.
- We have $\Gamma \vdash \Pi x_5\!:\!\{x_3\}_P.x_5 \triangleright_{\delta} \Pi x_5\!:\!\{x_3\}_P.x_3$ since $\Gamma; x_5 : \{x_3\}_P \vdash x_5\ \delta\ x_3$.

2.3 Typing

The typing rules of $\lambda^{\beta\delta}_{\leq\{\}}$ are given Figure 3. Four kinds of judgments are used:

- Context formation $\Gamma \vdash \mathsf{ok}$
- Type formation $\Gamma \vdash A$
- Subtyping $\Gamma \vdash A_1 \leq A_2$
- Typing $\Gamma \vdash M : A$

Let us give some explanations for these rules:

- We choose not to explicitly check the well-formedness of contexts in order to give a presentation closer to the type-checking algorithm. Therefore, contrasting with a more traditional presentation, the rule T/VAR does not check the well-formedness of the context whereas the rules TY/PROD and T/LAM check that the domain A_1 is a well-formed type. Similarly, the rules for subtyping do not ensure the upper type is well-formed, whence the premise $\Gamma \vdash A$ for rule T/SUB.
- T/STR is a rule to strengthen the type of a term: whenever M has type A, it has also type $\{M\}_A$. Such a rule is quite natural, but is not a consequence of the other rules. Thus it allows to derive $x : P \vdash x : \{x\}_P$, which could not be derived otherwise.
- SUB/SINGR is the only rule introducing a singleton on the right of a subtyping judgment: a type can be lower than a singleton type $\{M_2\}_{A_2}$ only if it is itself a singleton type less than A_2 and whose contents is equal to M_2.
- SUB/SINGL introduces a singleton on the left of a subtyping judgment: a singleton $\{M_1\}_{A_1}$ is less than any type greater than A_1. When trying to check that a singleton is less than a given type, the rule SUB/SINGL may prove too coarse. Indeed, it completely forgets the information that the only element of $\{M_1\}_{A_1}$ is M_1. This can be problematic if A_1 is a product:

Subtyping

$$\mathsf{SUB/SET}\frac{}{\Gamma \vdash P \leq P} \qquad \mathsf{SUB/PROD}\frac{\Gamma \vdash A_1' \leq A_1 \qquad \Gamma; x : A_1' \vdash A_2 \leq A_2'}{\Gamma \vdash \Pi x\!:\!A_1.A_2 \leq \Pi x\!:\!A_1'.A_2'}$$

$$\mathsf{SUB/SINGR}\frac{\Gamma \vdash M_1 \bowtie M_2 \qquad \Gamma \vdash \{M_1\}_{A_1} \leq A_2}{\Gamma \vdash \{M_1\}_{A_1} \leq \{M_2\}_{A_2}}$$

$$\mathsf{SUB/SINGL}\frac{\Gamma \vdash A_1 \leq A_2}{\Gamma \vdash \{M_1\}_{A_1} \leq A_2} \qquad \mathsf{SUB/SINGPROD}\frac{\Gamma \vdash \Pi x\!:\!A_1.\{(M_1\ x)\}_{A_2} \leq A}{\Gamma \vdash \{M_1\}_{\Pi x A_1.A_2} \leq A}$$

Typing

$$\mathsf{T/VAR}\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \mathsf{T/LAM}\frac{\Gamma \vdash A_1 \qquad \Gamma; x : A_1 \vdash M : A_2}{\Gamma \vdash \lambda x\!:\!A_1.M : \Pi x\!:\!A_1.A_2}$$

$$\mathsf{T/APP}\frac{\Gamma \vdash M_1 : \Pi x\!:\!A_1.A_2 \qquad \Gamma \vdash M_2 : A_1}{\Gamma \vdash (M_1\ M_2) : A_2\{x \leftarrow M_2\}}$$

$$\mathsf{T/STR}\frac{\Gamma \vdash M : A}{\Gamma \vdash M : \{M\}_A}$$

$$\mathsf{T/SUB}\frac{\Gamma \vdash M : A' \qquad \Gamma \vdash A \qquad \Gamma \vdash A' \leq A}{\Gamma \vdash M : A}$$

Well-formed types:

$$\mathsf{TY/SET}\frac{}{\Gamma \vdash P} \qquad \mathsf{TY/SING}\frac{\Gamma \vdash M : A}{\Gamma \vdash \{M\}_A}$$

$$\mathsf{TY/PROD}\frac{\Gamma \vdash A_1 \qquad \Gamma; x : A_1 \vdash A_2}{\Gamma \vdash \Pi x\!:\!A_1.A_2}$$

Well-formed environments:

$$\mathsf{E/EMPTY}\frac{}{\vdash \mathsf{ok}} \qquad \mathsf{E/ADD}\frac{\Gamma \vdash \mathsf{ok} \qquad \Gamma \vdash A}{\Gamma; x : A \vdash \mathsf{ok}}$$

Fig. 3. Typing rules for $\lambda^{\beta\delta}_{\leq\{\}}$

in order to conclude

(1) $$\Gamma \vdash \{\lambda x\!:\!P.x\}_{\Pi x P.P} \leq \Pi x\!:\!P.\{x\}_P$$

SUB/SINGL requires the precondition $\Gamma \vdash (\lambda x\!:\!P.x) \leq \Pi x\!:\!P.\{x\}_P$, which is not derivable. Therefore, we introduce a new rule SUB/SINGPROD, which propagates the information that, for any product A_1 the domain of the singleton $\{M_1\}_{A_1}$ is itself a singleton: in order to derive the judgment 1, with SUB/SINGPROD, one has to derive the precondition

$$\Gamma \vdash \Pi x\!:\!P.\{((\lambda x\!:\!P.x)\ x)\}_P \leq \Pi x\!:\!P.\{x\}_P$$

which can easily be derived.

3 Metatheory

In this section, we sketch the metatheory of $\lambda^{\beta\delta}_{\leq\{\}}$. We aim at proving the subject-reduction, Church-Rosser and strong normalization properties. These results notably allow to implement the convertibility test over typed terms needed by the type-checking algorithm given Section 4.

Section 3.1 reviews the usual approaches to these issues and explains why the δ-reduction makes them fail. Section 3.2 introduces a new syntactic construct and a decomposition of the δ-reduction into two new reductions, the δ' and c reductions, enjoying better properties with respect to substitutions. Then, Section 3.3 sketches a proof of subject-reduction, Church-Rosser property and strong normalization.

3.1 Bad Behavior of the δ-reduction on Untyped Terms

3.1.1 Substitution and Reduction

Usual proofs of subject-reduction comprise several steps, one of which is the substitution property, stating that judgments are preserved by well-typed substitutions. In usual systems with dependent types, the proof of this property involves a lemma stating that the convertibility is preserved by substitution. Proving this lemma is trivial when the conversion is the β-equivalence, as the β-reduction is preserved by (untyped) substitutions: for any M_1, M_2, x, and M, $M_1 \triangleright_\beta M_2$ implies $M_1\{x \leftarrow M\} \triangleright_\beta M_2\{x \leftarrow M\}$.

One would like to have a similar property telling that for any context $\Gamma, x : A, \Delta$, any M_1, M_2, and M, $\Gamma, x : A, \Delta \vdash M_1 \triangleright_\delta M_2$ implies $\Gamma, \Delta\sigma \vdash M_1\sigma \triangleright_\delta M_2\sigma$ with $\sigma = \{x \leftarrow M\}$, but this property does not hold.

Consider for instance

$$\Gamma = x_1 : P; x_2 : P; x_3 : \{x_1\}_P \quad \text{and} \quad \Delta = \epsilon$$

Then $\Gamma; x : \{x_3\}_P; \Delta \vdash x \triangleright_\delta x_3$. Now, consider the substitution $\sigma = \{x \leftarrow x_2\}$; we do not have $\Gamma; \Delta\sigma \vdash x\sigma \triangleright_\delta x_3\sigma$.

One may think that the substitution property for δ-reduction should however hold if one requires the substitution to be well-typed, that is that x be substituted by a term of type $\{x_3\}_P$. But this is false: consider x_1; thanks to the rule T/STR, it has type $\{x_1\}_P$; thanks to the rule T/SUB, it also has type $\{x_3\}_P$; let $\sigma' = \{x \leftarrow x_1\}$; we do not have $\Gamma \vdash x\sigma' \triangleright_\delta x_3\sigma'$, but instead $\Gamma \vdash x_3\sigma' \triangleright_\delta x\sigma'$.

One could hope to prove the weaker property that convertibility is preserved by well-typed substitution. But proving it seems to involve some subtle arguments depending on the type of the variable being substituted, and on the interaction between δ-rules and subtyping.

3.1.2 Proving the Church-Rosser Property

The Church-Rosser property for lambda-calculi with only β-reduction is generally proved on untyped terms using the Tait-Martin-Löf method as described in [9]. Unfortunately, as we show below, the Church-Rosser property for $\triangleright_{\beta\delta}$ does not hold for untyped terms in $\lambda^{\beta\delta}_{\leq\{\}}$.

A priori, this does not preclude us from using such a proof method for proving the Church-Rosser property on untyped terms. Indeed, in his study of $\beta\eta$-reduction for the Calculus of Constructions [6], Geuvers shows the Church-Rosser property holds up to the erasure of types on lambda-abstractions, using the argument that $\beta\eta$ is Church-Rosser for the type-free lambda-calculus. This weaker property is enough to show the subject-reduction; then the strong normalization property can be proved as well. Then, Geuvers shows that terms having the same type and equal up to the erasure of types on lambda-abstractions have a common $\beta\eta$-normal form, hence the Church-Rosser property.

Unfortunately, the case of $\lambda^{\beta\delta}_{\leq\{\}}$ is worse: whereas for $\beta\eta$ the critical pairs can be closed up to the erasure of types on lambda-abstractions, in $\lambda^{\beta\delta}_{\leq\{\}}$ the members of a critical pair can be arbitrarily different. In fact, for any pair of terms (M_1, M_2), there exists an untyped term M such that M $\beta\delta$-reduces to M_1 and $\beta\delta$-reduces to M_2:

$$M = ((\lambda x{:}\{M_1\}_P.x)\ M_2)$$

is such a term. Indeed, we have $M \triangleright_\beta M_2$ as M is a β-redex, and we have also

$$\vdash M \triangleright_\delta ((\lambda x{:}\{M_1\}_P.M_1)\ M_2) \triangleright_\beta M_1$$

3.2 Coercions

We analyze the lack of properties of δ-reduction with respect to substitution as follows: when a variable x declared of type A is substituted by a term M, its pre-principal type changes; the original is lost and some reductions may therefore be lost also. As we want to keep these reductions, we have to keep the information that the occurrences of M come from the substitution of x of type A. Therefore, we introduce a new syntactic construct $(M : A)$ building a term from any term M and any type A, called coercion. We introduce a new δ-rule for coercions:

$$\mathsf{D/COER}\frac{}{\Gamma \vdash_{pp} (M : A) : A}$$

as well as a new typing rule:

$$\mathsf{T/COER}\frac{\Gamma \vdash M : A}{\Gamma \vdash (M : A) : A}$$

We also define a new reduction relation, $\triangleright_c$, called c-reduction, or coercion removal, defined as the least monotonic relation such that $(M : A) \triangleright_c M$.

Then, we can prove the following restricted substitution property for the δ-reduction:

Proposition 3.1 (Restricted Substitution Property for δ-reduction) *For any variable x, any contexts Γ and Δ, any terms M, M_1, and M_2, and any types A and A', let $\sigma = \{x \leftarrow (M : A)\}$, then*

- *if $\Gamma, x : A, \Delta \vdash_{pp} M_1 : A'$ then*

$$\Gamma, \Delta\sigma \vdash_{pp} M_1\sigma : A'\sigma$$

- *if $\Gamma, x : A, \Delta \vdash M_1 \rhd_\delta M_2$ then*

$$\Gamma, \Delta\sigma \vdash M_1\sigma \rhd_\delta M_2\sigma$$

- *if $\Gamma \vdash M : A$ and $\Gamma, x : A, \Delta \vdash J$ where J is either ok, A', $A_1 \leq A_2$, $M' : A'$, or $M_1 \bowtie M_2$, then*

$$\Gamma, \Delta\sigma \vdash J\sigma$$

Proof. The proof is by induction on the definition of $\vdash_{pp}$ for the first property and by induction on the definition of δ-reduction for the second one. The third property is proved by induction on the derivation of the considered judgment. For the first property, remark the rule D/VAR is stable by restricted substitutions as the pre-principal type of $(M : A)$ is A, while not by unrestricted ones, which renders the substitution property false for unrestricted substitutions. Notice also that the first two properties do not need M to have type A nor even A to be well-typed since rule D/COER has no premise. □

However, δ-reduction still has an unexpected behavior. For instance, at some points in our metatheoretical development, one would like $\Gamma, x : A, \Delta \vdash_{pp} M_1 : A_1$ and $\Gamma \vdash A \rhd_\delta A'$ to imply the existence of A_2 such that $\Gamma, x : A', \Delta \vdash_{pp} M_1 : A_2$, and $\Gamma \vdash A_1 \rhd_\delta A_2$ or $A_1 = A_2$, but this property cannot be proved because the unrestricted substitution of rule D/APP is problematic.

Therefore, we slightly change the definition of δ-reduction. More precisely, we introduce a new relation called δ', such that $\Gamma \vdash M\ \delta'\ M'$ if there exists A such that $\Gamma \vdash_{pp'} M : \{M'\}_A$, where the judgment $\Gamma \vdash_{pp'} M : A$ is defined by the same rules as for $\Gamma \vdash_{pp} M : A$, except for the rule D/APP, which becomes:

$$\mathsf{D/APP}\frac{\Gamma \vdash_{pp'} M_1 : \Pi x\!:\!A_1.A_2}{\Gamma \vdash_{pp'} (M_1\ M_2) : A_2\{x \leftarrow (M_2 : A_1)\}}$$

Then, we define the δ'-reduction as the least monotonic relation such that $\Gamma \vdash M \rhd_{\delta'} M'$ whenever $\Gamma \vdash M\ \delta'\ M'$.

We also define a modified β-reduction relation, called β'-reduction, defined as the least monotonic relation such that for any x, A, M_1, and M_2, $(\lambda x\!:\!A.M_1\ M_2) \rhd_{\beta'} M_1\{x \leftarrow (M_2 : A)\}$.

Finally, we change the convertibility relation: $\Gamma \vdash M_1 \bowtie M_2$ is defined as

$$\exists M\ \Gamma \vdash M_1 \rhd^\star_{\beta'\delta'c} M \wedge \Gamma \vdash M_2 \rhd^\star_{\beta'\delta'c} M$$

As the β and δ-reductions are strategies for $\beta' c$ and $\delta' c$-reductions, being in the former notion of convertibility implies being in the latter. The converse is also true on typed terms, and is a consequence of the strong normalization and Church-Rosser properties proved in the next section.

3.3 Normalization

We prove the Subject-Reduction, Church-Rosser and Strong Normalization properties simultaneously, following an idea proposed by Goguen [7] for the Calculus of Constructions with $\beta\eta$-reduction. Our proof is inspired by Coquand and Gallier's proofs [5,3].

3.3.1 Elementary Properties of $\lambda^{\beta\delta}_{\leq\{\}}$

We first give a few elementary properties which will be useful in the following sections.

Lemma 3.2 *For any context Γ and any type A, $\Gamma \vdash A \leq A$ holds.*

Lemma 3.3 *For any context Γ, and any types A_1 and A_2 such that $\Gamma \vdash A_1 \triangleright_{\beta'\delta' c} A_2$, then $\Gamma \vdash A_1 \leq A_2$ and $\Gamma \vdash A_2 \leq A_1$ hold.*

Proof. Both Lemmas 3.2 and 3.3 are proved by induction on the considered types. □

Definition 3.4 (Context extension) *The context extension relation, noted $\supseteq$ is the smallest reflexive and transitive binary relation such that for any context Γ, Δ, any variable x and any type A, we have $\Gamma, x : A, \Delta \supseteq \Gamma, \Delta$.*

$\supseteq$ is obviously an ordering relation.

Proposition 3.5 (Weakening) *For any contexts Γ, Δ, any variable x, any type A, and any J being $A_1 \leq A_2$, A_1, or $M_1 : A_1$, if $\Gamma, \Delta \vdash J$ holds, then $\Gamma, x : A, \Delta \vdash J$ holds. Moreover, if $\Gamma, \Delta \vdash \mathsf{ok}$ holds and $\Gamma \vdash A$ holds, then $\Gamma, x : A, \Delta \vdash \mathsf{ok}$ holds. As a consequence, for any contexts Γ and Γ' such that $\Gamma' \supseteq \Gamma$, any J being $A_1 \leq A_2$, A_1, or $M_1 : A_1$, if $\Gamma \vdash J$ holds, then $\Gamma' \vdash J$ holds.*

Proof. By induction on the derivation of the considered judgment. □

3.3.2 Interpretations of Types and Contexts

The idea of the normalization proof is to interpret types as sets of terms such that every term belong to the interpretation of its types, and every interpretation contain only normalizing terms. More precisely, we define interpretations such that they contain only *semantic objects*, that is, well-typed normalizing terms having a unique normal form and whose type is preserved by reduction. Thus we prove subject-reduction, Church-Rosser and strong normalization at once. As we want the interpretation of convertible types to be equal, we interpret only *semantic types*, that is, well-formed normalizing types having a

unique normal form and whose well-formedness is preserved by reduction. We now give the formal definitions of semantic objects and semantic types.

Definition 3.6 (Semantic Types) *Given a context Γ, the set ST_Γ of semantic types in Γ is the smallest set of types such that any type A fulfilling the following conditions belongs to ST_Γ:*

(i) *for any A' such that $\Gamma \vdash A \triangleright_{\beta'\delta'c} A'$, $A' \in ST_\Gamma$,*

(ii) *and A has exactly one $\beta'\delta'c$ normal form,*

(iii) *and $\Gamma \vdash A$ is derivable.*

Remark 3.7 *The first condition and the fact that ST_Γ is the smallest set fulfilling the enumerated conditions imply that all the elements of ST_Γ are strongly normalizing (otherwise the intersection of ST_Γ and the set of strongly normalizing types would be smaller and still verifying the three conditions). The second one implies that all elements of ST_Γ have the diamond property, and the third one implies that they all enjoy the subject-reduction property.*

Definition 3.8 (Semantic Objects) *Given a context Γ and a semantic type A in Γ, the set $SO_\Gamma(A)$ of semantic objects for A in Γ is the smallest set such any term M fulfilling the following conditions belongs to $SO_\Gamma(A)$:*

(i) *for any M' such that $\Gamma \vdash M \triangleright_{\beta'\delta'c} M'$, $M' \in SO_\Gamma(A)$,*

(ii) *and M has exactly one normal form,*

(iii) *and $\Gamma \vdash M : A$ is derivable.*

Remark 3.9 *$SO_\Gamma(A)$ contains only strongly normalizing terms enjoying the diamond and subject-reduction properties.*

Proposition 3.10 *For any context Γ and any two semantic types A_1 and A_2, if $\Gamma \vdash A_1 \leq A_2$ holds, then $SO_\Gamma(A_1) \subseteq SO_\Gamma(A_2)$.*

Proof. Consider $M \in SO_\Gamma(A_1)$, by induction on the reduction of M, it is enough to prove that $\Gamma \vdash M : A_2$ holds. Since A_2 is a semantic type, $\Gamma \vdash A_2$ holds, and the results follows from rule T/SUB. □

Proposition 3.11 *For any context Γ, and any types A_1 and A_2 such that $A_1 \in ST_\Gamma$ and $\Gamma \vdash A_1 \triangleright_{\beta'\delta'c} A_2$, then $A_2 \in ST_\Gamma$ and $SO_\Gamma(A_1) = SO_\Gamma(A_2)$.*

Proof. By Definition 3.6, Proposition 3.10, and Lemma 3.3. □

Definition 3.12 (Measure of Types) *We now define a measure ν on types as follows:*

$$\begin{aligned} \nu(P) &= 0 \\ \nu(\Pi x : A_1 . A_2) &= \nu(A_1) + \nu(A_2) + 1 \\ \nu(\{M\}_A) &= \nu(A) + 1 \end{aligned}$$

This measure is clearly invariant by substitution as there is no type variable in $\lambda^{\beta\delta}_{\leq\{\}}$: for any variable x, any type A and any term M, $\nu(A\{x \leftarrow M\}) = \nu(A)$.

We can now define the interpretation of types by induction on this measure:

Definition 3.13 (Interpretations of Types) *The interpretation of a semantic type A in a context Γ, denoted by $[\![A]\!]_\Gamma$, is defined as follows:*

- $[\![P]\!]_\Gamma = SO_\Gamma(P)$;
- $[\![\Pi x\!:\!A_1.A_2]\!]_\Gamma = \{M \in SO_\Gamma(\Pi x\!:\!A_1.A_2) \mid \forall \Gamma' \supseteq \Gamma\ \forall M' \in [\![A_1]\!]_{\Gamma'}\ (M\ M') \in [\![A_2\{x \leftarrow (M' : A_1)\}]\!]_{\Gamma'}\}$;
- $[\![\{M\}_A]\!]_\Gamma$ *is the set of elements* $[\![A]\!]_\Gamma$ *convertible to* M *if* $M \in [\![A]\!]_\Gamma$, *and the empty set otherwise.*

Definition 3.14 (Acceptable Types) *The set ACC_Γ of* acceptable types *in a context Γ, is defined by induction on the measure of types as follows:*

- $P \in ACC_\Gamma$;
- $\Pi x\!:\!A_1.A_2 \in ACC_\Gamma$ *if and only if* $\Pi x\!:\!A_1.A_2 \in ST_\Gamma$, $A_1 \in ACC_\Gamma$, *and for any* $\Gamma' \supseteq \Gamma$, *any* $M \in [\![A_1]\!]_{\Gamma'}$, *we have* $A_2\{x \leftarrow (M : A_1)\} \in ACC_\Gamma$;
- $\{M\}_A \in ACC_\Gamma$ *if and only if* $\{M\}_A \in ST_\Gamma$, *and* $A \in ACC_\Gamma$, *and* $M \in [\![A]\!]_\Gamma$.

Proposition 3.15 *for any context Γ, any acceptable type A, and any terms M_1 and M_2, if $\Gamma \vdash M_1 \rhd_{\beta'\delta'c} M_2$ and $M_1 \in [\![A]\!]_\Gamma$, then $M_2 \in [\![A]\!]_\Gamma$.*

Proof. By induction on A. □

Proposition 3.16 *For any context Γ, any types A_1 and A_2, if $\Gamma \vdash A_1 \rhd_{\beta'\delta'c} A_2$ and $A_1 \in ACC_\Gamma$, then $[\![A_1]\!]_\Gamma = [\![A_2]\!]_\Gamma$ and $A_2 \in ACC_\Gamma$.*

Proof. By induction on A_1 and case inspection, using Lemmas 3.2 and 3.3.□

Definition 3.17 (Interpretation of a Context) *The interpretation $[\![\Gamma']\!]_\Gamma$ of a context Γ' into a context Γ is the set of substitutions over the variables appearing in Γ' defined as follows:*

- $[\![\epsilon]\!]_\Gamma = \{\sigma_{id}\}$ *where σ_{id} denotes the substitution mapping any variable to itself.*
- $[\![\Gamma''; x : A]\!]_\Gamma = \{\sigma + \{x \leftarrow (M : A\sigma)\} \mid \sigma \in [\![\Gamma'']\!]_\Gamma$ *and* $M \in [\![A\sigma]\!]_\Gamma$ *and* $A\sigma \in ACC_\Gamma\}$

Proposition 3.18 (Interpretations grow with the context) *For any Γ and Γ' such that $\Gamma' \supseteq \Gamma$, the following properties hold:*

- $ST_\Gamma \subseteq ST_{\Gamma'}$;
- *for any* $A \in ST_\Gamma$, $SO_\Gamma(A) \subseteq SO_{\Gamma'}(A)$;
- *for any* $A \in ST_\Gamma$, $[\![A]\!]_\Gamma \subseteq [\![A]\!]_{\Gamma'}$;
- $ACC_\Gamma \subseteq ACC_{\Gamma'}$.
- $[\![\Gamma'']\!]_\Gamma \subseteq [\![\Gamma'']\!]_{\Gamma'}$.

3.3.3 Saturation Properties

Definition 3.19 (Pre-Principal Type) *We define the* pre-principal type *of M in Γ as the unique A such that $\Gamma \vdash_{pp'} M : A$ if it exists and $\bot$ otherwise.*

Definition 3.20 (First Kind of Neutral Terms) *Let Γ be a context and $A \in ACC_\Gamma$. We define the set $N^n_\Gamma(A)$ of* first kind of neutral terms of level n for the type A in Γ *by induction on n as follows:*

- $N^0_\Gamma(A) = \emptyset$;
- $N^{n+1}_\Gamma(A)$ *is the set of terms M such that the following conditions hold:*

(i) $\Gamma \vdash M : A$ *holds;*
(ii) *M is not a lambda-abstraction;*
(iii) *$PP_\Gamma(M)$ is $\bot$ or a singleton type;*
(iv) *there exists $M_0 \in [\![A]\!]_\Gamma$ such that $\Gamma \vdash M \rhd^\star_{\beta'\delta'c} M_0$.*
(v) *all terms M' verifying $\Gamma \vdash M \rhd_{\beta'\delta'c} M'$ belong to $[\![A]\!]_\Gamma \cup N^n_\Gamma(A)$ and convert with M_0.*

The set $N_\Gamma(A)$ of first kind of neutral terms *for the type A in Γ is defined as $\bigcup_{n\in\mathbb{N}} N^n_\Gamma(A)$.*

Definition 3.21 (Second Kind of Neutral Terms) *Let Γ be a context and $A \in ACC_\Gamma$. We define the set $N'^n_\Gamma(A)$ of* second kind of neutral terms of level n for the type A in Γ *by induction on n as follows:*

- $N'^0_\Gamma(A) = \emptyset$;
- $N'^{n+1}_\Gamma(A)$ *is the set of terms M such that the following conditions hold:*

(i) $\Gamma \vdash M : A$ *holds;*
(ii) *M is not a lambda-abstraction;*
(iii) *$PP_\Gamma(M)$ is an acceptable type and converts with A.*
(iv) *all terms M' verifying $\Gamma \vdash M \rhd_{\beta'\delta'c} M'$ belong to $[\![A]\!]_\Gamma \cup N'^n_\Gamma(A)$ and convert one with each other;*

The set $N'_\Gamma(A)$ of second kind of neutral terms *for the type A in Γ is defined as $\bigcup_{n\in\mathbb{N}} N'^n_\Gamma(A)$.*

Proposition 3.22 *For any context Γ, any $A \in ACC_\Gamma$ and any $n \in \mathbb{N}$, $N^n_\Gamma(A) \subseteq N^{n+1}_\Gamma(A)$ and $N'^n_\Gamma(A) \subseteq N'^{n+1}_\Gamma(A)$.*

Proof. By induction on $n \in \mathbb{N}$. □

Proposition 3.23 *For any context Γ, any $A \in ACC_\Gamma$, $N_\Gamma(A) \subseteq SO_\Gamma(A)$ and $N'_\Gamma(A) \subseteq SO_\Gamma(A)$.*

Proof. By induction on n, we show that $N^n_\Gamma(A) \subseteq SO_\Gamma(A)$ and $N'^n_\Gamma(A) \subseteq SO_\Gamma(A)$. □

Proposition 3.24 *For any context Γ, any acceptable types A_1 and A_2, if $\Gamma \vdash A_1 \rhd_{\beta'\delta'c} A_2$, then $N_\Gamma(A_1) = N_\Gamma(A_2)$ and $N'_\Gamma(A_1) = N'_\Gamma(A_2)$.*

Proof. By induction on n, we prove $N^n_\Gamma(A_1) = N^n_\Gamma(A_2)$ and $N'^n_\Gamma(A_1) = N'^n_\Gamma(A_2)$ using Lemma 3.3 and Proposition 3.16. □

Lemma 3.25 *For any context Γ, any variable x, any types A_1 and A_2, any term $M \in N_\Gamma(\Pi x : A_1.A_2)$, any term $M_1 \in [\![A_1]\!]_\Gamma$ we have $(M\ M_1) \in N_\Gamma(A_2\{x \leftarrow (M_1 : A_1)\})$.*

Proof. By induction on the reduction of M and M_1. The only difficult point is to prove the condition v of definition 3.20 holds. Since $PP_\Gamma(M)$ is a singleton or $\bot$, $PP_\Gamma((M\ M_1)) = \bot$, which means that $(M\ M_1)$ is not a δ-redex. Since M is not a lambda-abstraction, one step of reduction of $(M\ M_1)$ can therefore only lead to $(M'\ M_1)$ for $M' \in N_\Gamma(\Pi x : A_1.A_2) \cup [\![\Pi x\!:\!A_1.A_2]\!]_\Gamma$ with $\Gamma \vdash M \rhd_{\beta'\delta'c} M'$ or to $(M\ M_1')$ with $\Gamma \vdash M_1 \rhd_{\beta'\delta'c} M_1'$. □

Lemma 3.26 *For any context Γ, any type $A \in ACC_\Gamma$, and any $M \in A$, $N_\Gamma(\{M\}_A) \subseteq N_\Gamma(A)$.*

Proof. By induction on n, we prove $N^n_\Gamma(\{M\}_A) \subseteq N^n_\Gamma(A)$. □

Proposition 3.27 *For any context Γ, any $A \in ACC_\Gamma$, $N_\Gamma(A) \subseteq [\![A]\!]_\Gamma$.*

Proof. By induction on A:

- $N_\Gamma(P) \subseteq [\![P]\!]_\Gamma$ since $N_\Gamma(P) \subseteq SO_\Gamma(P)$ and $[\![P]\!]_\Gamma = SO_\Gamma(P)$ by Definition 3.13.
- If $A = \Pi x\!:\!A_1.A_2$, then consider $M \in N_\Gamma(A)$. Let $\Gamma' \supseteq \Gamma$ and $M_1 \in [\![A_1]\!]_{\Gamma'}$. Then $M \in N_{\Gamma'}(A)$, therefore by Lemma 3.25 $(M\ M_1) \in N_{\Gamma'}(A_2\{x \leftarrow (M_1 : A_1)\})$. By induction hypothesis, $(M\ M_1) \in [\![A_2\{x \leftarrow (M_1 : A_1)\}]\!]_\Gamma$. Hence $M \in [\![A]\!]_\Gamma$.
- If $A = \{M_1\}_{A_1}$, then consider $M \in N_\Gamma(A)$. By Lemma 3.26, $M \in N_\Gamma(A_1)$. Therefore $M \in [\![A_1]\!]_\Gamma$. Moreover, there exists $M_0 \in [\![A]\!]_\Gamma$ such that $\Gamma \vdash M \rhd^\star_{\beta'\delta'c} M_0$. Therefore, the normal form of M, M_0, and M_1 is the same, so $M \in [\![A]\!]_\Gamma$.

□

Lemma 3.28 *For any context Γ, any $A \in ACC_\Gamma$, any $M \in [\![A]\!]_\Gamma$,*

$$N'_\Gamma(\{M\}_A) \subseteq N_\Gamma(\{M\}_A)$$

Proof. By definition of neutral terms, taking M as the M_0 needed in conditions iv and v of definition 3.20. □

Lemma 3.29 *For any context Γ, any variable x, any types A_1 and A_2, any term $M \in N'_\Gamma(\Pi x : A_1.A_2)$, any term $M_1 \in [\![A_1]\!]_\Gamma$ we have $(M\ M_1) \in N'_\Gamma(A_2\{x \leftarrow (M_1 : A_1)\})$.*

Proof. Similar to the proof of lemma 3.25. □

Proposition 3.30 *For any context Γ, any $A \in ACC_\Gamma$, $N'_\Gamma(A) \subseteq [\![A]\!]_\Gamma$.*

Proof. By induction on A, using Proposition 3.27 as well as Lemma 3.28 and Lemma 3.29. □

The usual saturation lemma about β-redexes can be proved for β'-redexes:

Lemma 3.31 (Saturation for β'-redexes) *For any context Γ, for any acceptable types A, A_1, and A_2, for any terms M_1 and M_2, for any variable x, if $\lambda x : A.M_1 \in SO_\Gamma(\Pi x : A_1.A_2)$, $M_2 \in SO_\Gamma(A_1)$ and $M_1\{x \leftarrow (M_2 : A)\} \in [\![A_2\{x \leftarrow (M_2 : A)\}]\!]_\Gamma$, then $(\lambda x : A.M_1\ M_2) \in N_\Gamma(A_2\{x \leftarrow (M_2 : A)\})$. As a corollary, $(\lambda x : A.M_1\ M_2) \in [\![A_2\{x \leftarrow (M_2 : A)\}]\!]_\Gamma$.*

Proof. By induction on the reduction of $\lambda x : A.M_1$ and M_2, using Proposition 3.24. □

Lemma 3.32 (Saturation for Variables) *For any context Γ, any acceptable type A in Γ, any variable x declared of type A in Γ belongs to $N'_\Gamma(A)$ and therefore to $[\![A]\!]_\Gamma$.*

Proof. x belongs to $N'^1_\Gamma(A)$: even if x is not in normal form, then A is a singleton type $\{M'\}_{A'}$ and since $A \in ACC_\Gamma$, $M' \in [\![A']\!]_\Gamma$. □

Lemma 3.33 (Saturation for c-redexes) *For any context Γ, any acceptable type A in Γ, any $M \in [\![A]\!]_\Gamma$, $(M : A)$ belongs to $N'_\Gamma(A)$ and therefore to $[\![A]\!]_\Gamma$.*

Proof. By induction on the reduction of A and M, using Proposition 3.24 and 3.16. □

3.3.4 Main Proof

Thanks to the saturations proofs, the following main lemma can be proved easily:

Lemma 3.34 *For any contexts Γ and Γ' and any $\sigma \in [\![\Gamma']\!]_\Gamma$ we have the following properties:*

- *If $\Gamma' \vdash M_1 \bowtie M_2$ then $\Gamma \vdash M_1\sigma \bowtie M_2\sigma$.*
- *If $\Gamma' \vdash A$ then $A\sigma \in ACC_\Gamma$.*
- *If $\Gamma' \vdash A_1 \leq A_2$, $A_1\sigma \in ACC_\Gamma$, and $A_2\sigma \in ACC_\Gamma$ then $[\![A_1\sigma]\!]_\Gamma \subseteq [\![A_2\sigma]\!]_\Gamma$.*
- *If $\Gamma' \vdash M : A$ then $A\sigma \in ACC_\Gamma$ and $M \in [\![A\sigma]\!]_\Gamma$.*

Proof. The proof is performed by induction on the considered derivation. □

As a corollary we can prove

Lemma 3.35 *For any context Γ, if $\Gamma \vdash \mathsf{ok}$, then $\sigma_{id} \in [\![\Gamma]\!]_\Gamma$.*

Proof. The proof is by induction on the derivation of $\Gamma \vdash \mathsf{ok}$, using Lemmas 3.34 and 3.33. □

Theorem 3.36 *For any context Γ such that $\Gamma \vdash \mathsf{ok}$, the following properties hold:*

- *If $\Gamma \vdash A$ then $A \in ACC_\Gamma$.*
- *If $\Gamma \vdash A_1 \leq A_2$, $A_1 \in ACC_\Gamma$, and $A_2 \in ACC_\Gamma$, then $[\![A_1]\!]_\Gamma \subseteq [\![A_2]\!]_\Gamma$. As a corollary, if $\Gamma \vdash A_1 \leq A_2$, $\Gamma \vdash A_1$, and $\Gamma \vdash A_2$, then $[\![A_1]\!]_\Gamma \subseteq [\![A_2]\!]_\Gamma$.*

- *If* $\Gamma \vdash M : A$ *then* $A \in ACC_\Gamma$ *and* $M \in [\![A]\!]_\Gamma$.

Proof. By Lemmas 3.34 and 3.35. □

As a consequence all typed terms in a well-formed context are semantic objects, *i.e.*, they have the subject-reduction property, have a unique normal form and are strongly normalizing.

4 Typing Algorithm

Our type-inference and type-checking algorithms for $\lambda^{\beta\delta}_{\leq\{\}}$ are based on the following seven judgments:

- $\Gamma \vdash_c A_1 \leq A_2$, checking A_1 is a subtype of A_2 in Γ (assuming A_1 and A_2 are well-formed types and Γ is well-formed);
- $\Gamma \vdash_{qp} M : A$, inferring the quasi-principal type A of M in Γ (assuming Γ is well-formed);
- $\Gamma \vdash_p M : A$, inferring the principal type A of M in Γ (assuming Γ is well-formed);
- $\Gamma \vdash_c M : A$, checking A is a well-formed type and M has type A in Γ (assuming Γ is well-formed).
- $\Gamma \vdash_c^{wft} M : A$, checking M has type A in Γ (assuming A is a well-formed type and Γ is well-formed).
- $\Gamma \vdash_c A$, checking the type A is well-formed (assuming Γ is well-formed).
- $\Gamma \vdash_c \mathsf{ok}$, checking the environment Γ is well-formed.

Rules for these judgments are given Figure 4. They are syntax-directed. The preconditions over the judgment $\Gamma \vdash_c A_1 \leq A_2$ imply that SUB/SINGR needs to decide convertibility between well-typed terms only, which can be done by normalization. Therefore, algorithms can straightforwardly be derived from these rules.

Notice that the rules for the algorithmic judgments are the same as the ones of Figure 3 up to the following differences:

- SUB/SINGPROD is restricted to the case SUB/SINGR does not apply;
- SUB/SINGL applies only when neither SUB/SINGR nor SUB/SINGPROD do;
- "$\vdash$" symbols appearing Figure 3 are now decorated with p, qp, c or c and wft.
- T/SUB has been split into two rules: T/SUBPRE and T/SUB2.

Proposition 4.1 (Soundness of the Algorithms) *The rules of Figure 4 are sound. More precisely:*

- *If* $\Gamma \vdash_c A_1 \leq A_2$, *then* $\Gamma \vdash A_1 \leq A_2$.
- *If* $\Gamma \vdash \mathsf{ok}$ *and*

Checking subtyping

$$\mathsf{SUB/SET}\frac{}{\Gamma \vdash_c P \leq P} \qquad \mathsf{SUB/PROD}\frac{\Gamma \vdash_c A_1' \leq A_1 \qquad \Gamma; x : A_1' \vdash_c A_2 \leq A_2'}{\Gamma \vdash_c \Pi x\!:\!A_1.A_2 \leq \Pi x\!:\!A_1'.A_2'}$$

$$\mathsf{SUB/SINGR}\frac{\Gamma \vdash M_1 \bowtie M_2 \qquad \Gamma \vdash_c \{M_1\}_{A_1} \leq A_2}{\Gamma \vdash_c \{M_1\}_{A_1} \leq \{M_2\}_{A_2}}$$

$$\mathsf{SUB/SINGL}\frac{\Gamma \vdash_c A_1 \leq A_2}{\Gamma \vdash_c \{M_1\}_{A_1} \leq A_2} A_2 \neq \{M\}_A, A_1 \neq \Pi x\!:\!A_3.A_4$$

$$\mathsf{SUB/SINGPROD}\frac{\Gamma \vdash_c \Pi x\!:\!A_1.\{(M_1\ x)\}_{A_2} \leq A}{\Gamma \vdash_c \{M_1\}_{\Pi x A_1.A_2} \leq A} A \neq \{M_3\}_{A_3}$$

Quasi-principal type inference

$$\mathsf{T/VAR}\frac{\Gamma(x) = A}{\Gamma \vdash_{qp} x : A} \qquad \mathsf{T/LAM}\frac{\Gamma \vdash_c A_1 \qquad \Gamma; x : A_1 \vdash_{qp} M : A_2}{\Gamma \vdash_{qp} \lambda x\!:\!A_1.M : \Pi x\!:\!A_1.A_2}$$

$$\mathsf{T/APP}\frac{\Gamma \vdash_{qp} M_1 : \Pi x\!:\!A_1.A_2 \qquad \Gamma \vdash_c^{wft} M_2 : A_1}{\Gamma \vdash_{qp} (M_1\ M_2) : A_2\{x \leftarrow M_2\}}$$

Principal type inference

$$\mathsf{T/STR}\frac{\Gamma \vdash_{qp} M : A}{\Gamma \vdash_p M : \{M\}_A}$$

Type-checking a term in a well-formed type

$$\mathsf{T/SUBPRE}\frac{\Gamma \vdash_p M : A' \qquad \Gamma \vdash_c A' \leq A}{\Gamma \vdash_c^{wft} M : A}$$

Type-checking a term

$$\mathsf{T/SUB2}\frac{\Gamma \vdash_c A \qquad \Gamma \vdash_c^{wft} M : A}{\Gamma \vdash_c M : A}$$

Checking types

$$\mathsf{TY/SET}\frac{}{\Gamma \vdash_c P} \qquad \mathsf{TY/SING}\frac{\Gamma \vdash_c M : A}{\Gamma \vdash_c \{M\}_A}$$

$$\mathsf{TY/PROD}\frac{\Gamma \vdash_c A_1 \qquad \Gamma; x : A_1 \vdash_c A_2}{\Gamma \vdash_c \Pi x\!:\!A_1.A_2}$$

Checking environments

$$\mathsf{E/EMPTY}\frac{}{\vdash_c \mathsf{ok}} \qquad \mathsf{E/ADD}\frac{\Gamma \vdash_c \mathsf{ok} \qquad \Gamma \vdash_c A}{\Gamma; x : A \vdash_c \mathsf{ok}}$$

Fig. 4. Type-checking and type inference algorithms

  - $\Gamma \vdash_{qp} M : A$,
  - *or* $\Gamma \vdash_c^{wft} M : A$ *and* $\Gamma \vdash A$,
  - *or* $\Gamma \vdash_p M : A$,
  - *or* $\Gamma \vdash_c M : A$,

  then $\Gamma \vdash M : A$.
- *If* $\Gamma \vdash \mathsf{ok}$ *and* $\Gamma \vdash_c A$, *then* $\Gamma \vdash A$.
- *If* $\Gamma \vdash_c \mathsf{ok}$, *then* $\Gamma \vdash \mathsf{ok}$.

Proof. The proof can be performed by a simple induction on the derivation of the judgment, since the rules for the algorithmic judgments mostly define a strategy for the rule for the non-algorithmic judgments. The only non-straightforward case is T/APP, which relies on the fact that its first premise implies A_1 is a well-formed type. □

Proposition 4.2 (Termination of the Algorithms) *The algorithms given Figure 4 terminate provided they are applied to arguments fulfilling the conditions associated to the judgments presented above.*

Proposition 4.3 (Completeness of the Algorithms) *The rules given Figure 4 are complete. More precisely:*

- *If* $\Gamma \vdash \mathsf{ok}$ *and* $\Gamma \vdash A_1 \leq A_2$ *and* $\Gamma \vdash A_1$ *and* $\Gamma \vdash A_2$ *then* $\Gamma \vdash_c A_1 \leq A_2$.
- *If* $\Gamma \vdash \mathsf{ok}$ *and* $\Gamma \vdash M : A$, *then*
  - $\Gamma \vdash_c M : A$,
  - *and* $\Gamma \vdash_c^{wft} M : A$,
  - *and there exists* A_1 *such that* $\Gamma \vdash_{qp} M : A_1$ *and* $\Gamma \vdash \{M\}_A \leq A_1$,
  - *and there exists* A_2 *such that* $\Gamma \vdash_p M : A_2$ *and* $\Gamma \vdash A_2 \leq A$.
- *If* $\Gamma \vdash \mathsf{ok}$ *and* $\Gamma \vdash A$ *then* $\Gamma \vdash_c A$.
- *If* $\Gamma \vdash \mathsf{ok}$ *then* $\Gamma \vdash_c \mathsf{ok}$.

Proof. The proof is by induction on the derivation of the involved judgment. It requires some additional lemmas such as the transitivity of subtyping. The lack of space prevents us to give them in details. □

5 Related Work

5.1 Reduction-based versus Rule-based Equality

Compared to the algorithm given by Harper and Stone [8], comparison of terms in $\lambda^{\beta\delta}_{\leq\{\}}$ is conceptually easy, as it only requires to $\beta\delta$-normalize them. It is also more flexible as one can choose any strategy.

The equality of terms in $\lambda^{\beta\delta}_{\leq\{\}}$ is an intentional equality: it is the smallest notion of equality compatible with reduction. In Harper and Stone's singletons [8] as well as in Aspinall's, equality of terms is parameterized by the type they are compared in. Their equality is more extensional than ours: given a context $\Gamma = x_1 : P$, whereas $\lambda x_2 : \{x_1\}_P . x_1$ and $\lambda x_2 : P . x_1$ are distinct $\beta\delta$-

normal form in $\lambda^{\beta\delta}_{\leq\{\}}$, in Aspinall's $\lambda_{\leq\{\}}$, they are equal at type $\Pi x_2 : \{x_1\}_P.P$. In general, identifying more terms is desirable; whether one can give a system that compares terms through $\beta\delta$-reduction and whose equality is more extensional than in $\lambda^{\beta\delta}_{\leq\{\}}$ is an open question.

5.2 Proof Method for Normalization

The proof method for subject-reduction, Church-Rosser property, and strong normalization is inspired by the one we developed in our thesis for a module calculus [4]. The idea to add coercions to the language to have the restricted substitution properties is especially useful.

The proof method of [4] is itself inspired by Goguen's thesis [7] which introduces a simultaneous proof of subject-reduction, Church-Rosser property and strong normalization of the Calculus of Constructions with $\beta\eta$-reduction. We identify the following ideas in Goguen's proof:

- Proving subject-reduction, Church-Rosser and strong normalization at once is slightly more difficult than proving the strong normalization property alone. It much simplifies the subject-reduction and Church-Rosser issues.
- Requiring the interpretations of types to contain only semantic objects in their definitions simplifies the proof. It replaces the need to prove that interpretations contains only semantic objects at a point where little is known about them to the need to prove that $(\lambda x : A.M)\sigma$ is a semantic object for the case of lambda-abstraction in the proof of Lemma 3.34, at a point where much more is known.
- Goguen defines a typed operational semantics, using the worst possible strategy for normalizing a term, to make the proof of the saturation lemmas easier.

We reused the first two of them for $\lambda^{\beta\delta}_{\leq\{\}}$ but we do not see how the reuse the third one. The main difficulty here is that $\lambda^{\beta\delta}_{\leq\{\}}$ has a subtyping notion; we could not see how to define a typed operational semantic taking into account this subtyping relation. However it seems that typed operational semantics can be used for higher-order subtyping [2]. Whether one can be given for singleton types is an interesting area for future work.

6 Conclusion

$\lambda^{\beta\delta}_{\leq\{\}}$ is a typed lambda-calculus with singleton types. Its equality notion is defined by convertibility through a new reduction called δ-reduction.

$\beta\delta$-reduction has a very odd behavior on untyped terms, as any pair of untyped terms has a common antecedent by $\triangleright^{\star}_{\beta\delta}$. As far as we know, $\beta\delta$ is the only reduction not defined on purpose enjoying such an odd behavior.

The usual metatheoretical properties could be proved though. This seems to show that Goguen's method for strong normalization is quite effective and

robust.

$\lambda^{\beta\delta}_{\leq\{\}}$ enjoys a straightforward type inference and type-checking algorithm. This algorithm just relies on a normalization function for $\beta\delta$-reduction, which is conceptually much simpler than the term comparison algorithm presented in [8].

Finally, whether the equality in $\lambda^{\beta\delta}_{\leq\{\}}$ can be made more extensional is an open question.

References

[1] David Aspinall. Subtyping with singleton types. In Leszek Pacholski and Jerzy Tiuryn, editors, *Proceedings of the 8th Workshop on Computer Science Logic*, volume 933 of *Lecture Notes in Computer Science*, pages 1–15, Kazimierz, Poland, September 1994. Springer-Verlag.

[2] Adriana Compagnoni and Healfdene Goguen. Typed operational semantics for higher order subtyping. Technical Report Technical Report ECS-LFCS-97-361, LFCS, University of Edinburgh, July 1997.

[3] Thierry Coquand. A meta-mathematical investigation of a Calculus of Constructions. Private Communication, 1987.

[4] Judicaël Courant. $\mathcal{MC}$: A module calculus for Pure Type Systems. Research Report 1217, LRI, June 1999.

[5] Jean Gallier. *Logic and Computer Science*, chapter On Girard's Candidats de Réductibilité. Academic Press, 1990. P. Odifreddi editor.

[6] Herman Geuvers. *Logics and Type Systems.* PhD thesis, University of Nijmegen, September 1993.

[7] Healfdene Goguen. *A Typed Operational Semantics for Type Theory.* PhD thesis, University of Edinburgh, Aug 1994. LFCS Report ECS-LFCS-94-304.

[8] Robert Harper and Christopher Stone. Deciding type equivalence with singleton kinds. In Thomas Reps, editor, *Conference Record of the 27th Symposium on Principles of Programming Languages*, pages 214–227, Boston, Masschusetts, January 2000. ACM Press.

[9] Masako Takahashi. Parallel reductions in λ-calculus. Technical report, Department of Information Science, Tokyo Institute of Technology, 1993. Internal report.