The Undecidability of Third Order Pattern Matching in Calculi with Dependent Types or Type Constructors

Gilles Dowek

L'Indécidabilité du Filtrage du Troisième Ordre dans les Calculs avec Types Dépendants ou Constructeurs de Types

Gilles Dowek

Résumé : On prouve l'indécidabilité du problème du filtrage du troisième ordre dans les λ-calcul typés avec types dépendants et dans ceux avec constructeurs de types en leur réduisant le problème de l'unification du second ordre.

The Undecidability of Third Order Pattern Matching in Calculi with Dependent Types or Type Constructors

Summary: We prove the undecidability of the third order pattern matching problem in typed λ-calculi with dependent types and in those with type constructors by reducing the second order unification problem to them.

Abridged English Version

We define eight typed λ-calculi, called the *calculi of the cube* [1]. All these calculi allow functions from terms to terms. Those that allow functions from terms to types are said to admit *dependent types*, those that allow functions from types to terms are said to be *polymorphic*, those that allow functions from types to types are said to admit *types constructors*. We write $Prop$ and $Type$ for the sorts written $*$ and $\Box$ in [1].

A *unification problem* is a triple $\langle \Gamma, a, b\rangle$ such that Γ is a context of universally and existentially quantified variables and a and b are well-typed terms of the same type in Γ. A *matching problem* is a unification problem such that b is closed in Γ. A substitution σ is said to be a *solution* of the problem $\langle \Gamma, a, b\rangle$ if it is well-typed in Γ and the terms σa and σb are equivalent.

A unification problem $\langle \Gamma, a, b\rangle$ is said to be *term-elementary* if and only if $\Gamma = [\forall U : Prop]\Gamma'$, for all declarations $Qx : T$ of Γ', the term T is one of the terms U, $U \to U$, $U \to U \to U$ or $U \to U \to U \to U$ and the common type of a and b is U. In a calculus with type constructors, a unification problem $\langle \Gamma, a, b\rangle$ is said to be *type-elementary*, if and only if for all declarations $Qx : T$ of Γ, the term T is one of the terms $Prop$, $Prop \to Prop$, $Prop \to Prop \to Prop$ or $Prop \to Prop \to Prop \to Prop$ and the common type of a and b is $Prop$.

Goldfarb's theorem [3] generalizes to all the calculi of the cube and shows that in all these calculi, there is no effective method that decides if a term-elementary unification problem has a solution and in the calculi with type constructors there is no effective method that decides if a type-elementary unification problem has a solution.

Theorem 1 *In a calculus with dependent types there is no effective method that decides if a matching problem in which all existential variables have at most third order types has a solution.*

Proof. For every term-elementary unification problem $\langle \gamma, u_1, u_2 \rangle$, we construct a matching problem $\langle \Gamma, t_1, t_2 \rangle$ such that all the types of the existential variables of Γ are at most third order and $\langle \gamma, u_1, u_2 \rangle$ has a solution if and only if $\langle \Gamma, t_1, t_2 \rangle$ also has one. We let:

$$\Gamma = \gamma[\forall z : U; \forall P : U \to Prop; \forall c : (P\ z); \forall d : (P\ z); \forall G : (P\ z) \to (P\ z) \to (P\ z);$$

$$\exists f : (h : U \to U)(P\ (h\ (u_1))) \to (P\ (h\ (u_2)))]$$

$$t_1 = (G\ (f\ [x : U]z\ c)\ (f\ [x : U]z\ d)) \qquad t_2 = (G\ c\ d)$$

If there exists a substitution τ such that $\tau u_1 = \tau u_2$ then we let:

$$\sigma = \tau \cup \{\langle f, [], [x_1 : U \to U][x_2 : (P\ (x_1\ (\tau u_1)))]x_2 \rangle\}$$

The substitution σ is well-typed in Γ and is a solution for the matching problem $\sigma t_1 = t_2$.

Conversely if there exists a substitution σ well-typed in Γ such that $\sigma t_1 = t_2$, then σ is well-typed in γ and σf must be the term $[x_1 : U \to U][x_2 : (P\ (x_1\ (\sigma u_1)))]x_2$, so $\sigma u_1 = \sigma u_2$.

Theorem 2 *In a calculus with type constructors there is no effective method that decides if a matching problem in which all existential variables have at most third order types has a solution.*

Proof. For every type-elementary unification problem $\langle \gamma, u_1, u_2 \rangle$ we build a matching problem $\langle \Gamma, t_1, t_2 \rangle$ such that all the types of existential variables of Γ are at most third order and $\langle \gamma, u_1, u_2 \rangle$ has a solution if and only if $\langle \Gamma, t_1, t_2 \rangle$ also has one. We let:

$$\Gamma = \gamma[\forall Z : Prop; \forall c : Z; \forall d : Z; \forall G : Z \to Z \to Z; \exists f : (h : Prop \to Prop)(h\ u_1) \to (h\ u_2)]$$

$$t_1 = (G\ (f\ [X : Prop]Z\ c)\ (f\ [X : Prop]Z\ d)) \qquad t_2 = (G\ c\ d)$$

If there exists a substitution τ such that $\tau u_1 = \tau u_2$ then we let:

$$\sigma = \tau \cup \{\langle f, [], [x_1 : Prop \to Prop][x_2 : (x_1\ (\tau u_1))]x_2 \rangle\}$$

The substitution σ is well-typed in Γ and is a solution of the matching problem $\sigma t_1 = t_2$.

Conversely if there exists a substitution σ well-typed in Γ such that $\sigma t_1 = t_2$, then σ is well typed in γ and σf must be the term $[x_1 : Prop \to Prop][x_2 : (x_1\ (\sigma u_1))]x_2$ so $\sigma u_1 = \sigma u_2$.

1 Le Cube des λ-calculs Typés

On définit huit λ-calculs typés appelés les *calculs du cube* [1]. Tous ces calculs autorisent les fonctions des termes dans les termes. Ceux qui autorisent les fonctions des termes dans les types sont dits admettre des *types dépendants*, ceux qui autorisent des fonctions des types dans les termes sont dits *polymorphes*, ceux qui autorisent des fonctions des types dans les types sont dits admettre des *constructeurs de types*. Un calcul peut avoir ou non chacune de ces trois propriétés.

Définition 1 (Syntaxe)

$$T \ ::= \ Prop \mid Type \mid x \mid (T\ T) \mid [x:T]T \mid (x:T)T$$

Dans cette note, on ignore les problèmes de capture de variables. Une présentation rigoureuse utiliserait des indices de de Bruijn. Un terme de la forme $[x:T]T'$ est appelé une λ-abstraction et un terme de la forme $(x:T)T'$ est appelé un produit. La notation $T \to T'$ est utilisée pour $(x:T)T'$ quand x n'a pas d'occurrence dans T'.

Soient t et t' deux termes et x une variable, on note $t[x \leftarrow t']$ le terme obtenu en substituant le terme t' à la variable x dans le terme t. On note $t = t'$ la relation de $\beta\eta$-équivalence c'est-à-dire la plus petite relation d'équivalence compatible avec la structure de terme qui vérifie :

$$(([x:T]t)u) = t[x \leftarrow u] \quad (\beta)$$

$$[x:T](t\ x) = t \ \textit{si}\ x \ \textit{n'est pas libre dans}\ t \quad (\eta)$$

Définition 2 *Un contexte est une liste de couples $x : T$ où x est une variable et T un terme.*

Définition 3 *Soit $R \subseteq \{\langle Prop, Prop\rangle, \langle Prop, Type\rangle, \langle Type, Prop\rangle, \langle Type, Type\rangle\}$ tel que $\langle Prop, Prop\rangle \in R$. On définit deux jugements : Γ est bien-formé et t a le type T dans Γ ($\Gamma \vdash t : T$) où Γ est un contexte et t et T deux termes :*

$$\frac{}{[]\ \textit{bien-formé}}$$

$$\frac{\Gamma \vdash T : s}{\Gamma[x:T]\ \textit{bien-formé}}\ s \in \{Prop, Type\}$$

$$\frac{\Gamma\ \textit{bien-formé}}{\Gamma \vdash Prop : Type}$$

$$\frac{\Gamma\ \textit{bien-formé} \quad x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$\frac{\Gamma \vdash T : s \quad \Gamma[x:T] \vdash T' : s'}{\Gamma \vdash (x:T)T' : s'}\ \langle s, s'\rangle \in R$$

$$\frac{\Gamma \vdash (x:T)T' : s \quad \Gamma[x:T] \vdash t : T'}{\Gamma \vdash [x:T]t : (x:T)T'}\, s \in \{Prop, Type\}$$

$$\frac{\Gamma \vdash t : (x:T)T' \quad \Gamma \vdash t' : T}{\Gamma \vdash (t\ t') : T'[x \leftarrow t']}$$

$$\frac{\Gamma \vdash T : s \quad \Gamma \vdash T' : s \quad \Gamma \vdash t : T \quad T = T'}{\Gamma \vdash t : T'}\, s \in \{Prop, Type\}$$

Le calcul tel que $R = \{\langle Prop, Prop\rangle\}$ est appelé λ-calcul simplement typé. Les calculs tels que $\langle Prop, Type\rangle \in R$ sont dits admettre des types dépendants, ceux tels que $\langle Type, Prop\rangle \in R$ sont dits polymorphes, ceux tels que $\langle Type, Type\rangle \in R$ sont dits admettre des constructeurs de types.

Définition 4 *Soient t et t' deux termes, la relation de $\beta\eta$-réduction $t \rhd t'$ est la plus petite relation réflexive, transitive et compatible avec la structure de terme qui vérifie :*

$$(([x:T]t)u) \rhd t[x \leftarrow u] \quad (\beta)$$

$$[x:T](t\ x) \rhd t \text{ si } x \text{ n'est pas libre dans } t \quad (\eta)$$

On conjecture que la relation de réduction sur les termes bien-typés est fortement normalisable et Church-Rosser. De ce fait, pour tout terme on peut définir une forme normale unique. Deux termes sont équivalents s'ils ont la même forme normale.

Proposition 1 *Un terme normal t est une abstraction, un produit ou un terme atomique c'est-à-dire un terme de la forme $(x\ t_1\ ...\ t_n)$ où x est une variable ou l'un des symboles $Prop$ et $Type$.*

2 Contextes Quantifiés

Définition 5 *Un Contexte Quantifié est un contexte Γ à chaque variable duquel on associe un quantificateur ($\forall$ ou $\exists$). Dans un contexte Γ, une variable à laquelle on associe le quantificateur universel (resp. existentiel) est dite universelle (resp. existentielle) dans ce contexte.*

Définition 6 *Un terme t bien-typé dans un context Γ est dit fermé dans Γ si pour tout x libre dans t, x est universelle dans Γ et son type est fermé dans le préfixe de Γ défini avant x.*

Définition 7 *Un ensemble fini σ de triplets $\langle x, \gamma, t\rangle$ où x est une variable γ un contexte dont toutes les variables sont existentielles et t est un terme est appelé une substitution si pour toute variable x il existe au plus un triplet de la forme $\langle x, \gamma, t\rangle$ dans σ.*

Définition 8 *Soit x une variable et σ une substitution. s'il y a un triplet $\langle x, \gamma, t\rangle$ dans σ on pose $\sigma x = t$, sinon on pose $\sigma x = x$. Cette définition s'étend aux termes de manière naturelle.*

Définition 9 *Soit un contexte quantifié Γ et une substitution σ. On définit, par récurrence sur la longueur de Γ, une relation de compatibilité : σ est bien-typée dans Γ, et si σ est bien-typée dans Γ, un contexte quantifié $\sigma\Gamma$.*
Si $\Gamma = []$ alors σ est bien-typée dans Γ et $\sigma\Gamma = []$.
Si $\Gamma = \Delta[\forall x : T]$ alors si σ est bien-typée dans Δ on pose $\Gamma' = \sigma\Delta$, si $\Gamma' \vdash \sigma T : Prop$ ou $\Gamma' \vdash \sigma T : Type$ alors σ est bien-typée dans Γ et $\sigma\Gamma = \Gamma'[\forall x : \sigma T]$. Dans les autres cas σ n'est pas bien-typée dans Γ.
Si $\Gamma = \Delta[\exists x : T]$ alors soit γ le contexte associé à x par σ si il existe un triplet $\langle x, \gamma, t\rangle$ dans σ et $\gamma = [\exists x : \sigma T]$ sinon. Si σ est bien-typée dans Δ on pose $\Gamma' = \sigma\Delta$. Si $\Gamma'\gamma$ est bien-formé et $\Gamma'\gamma \vdash \sigma x : \sigma T$ alors σ est bien-typée dans Γ et $\sigma\Gamma = \Gamma'\gamma$. Dans les autres cas σ n'est pas bien-typée dans Γ.

Définition 10 *Soit Γ un contexte quantifié et T un terme normal de type $Prop$ ou $Type$. Le terme T n'est pas une abstraction. L'ordre de T ($o(T)$) est l'élément de $N \cup \{\infty\}$ défini par :*

Si T est atomique $T = (x\ t_1\ ...\ t_n)$ alors si x est une variable universelle de Γ et on pose $o(T) = 1$, si x est une variable existentielle de Γ on pose $o(T) = \infty$, si $x = Prop$ on pose $o(T) = 2$.

Si T est un produit $T = (y : U)V$ alors on pose $o(T) = max\{1 + u, v\}$ où u est l'ordre de U dans le contexte quantifié Γ et v est l'ordre de V dans le contexte quantifié $\Gamma[\exists y : U]$.

Définition 11 *Soit Γ un contexte et a et b deux termes bien typés et de même type dans Γ. Le triplet $\langle \Gamma, a, b\rangle$ est appelé problème d'unification. Si le terme b est fermé dans Γ il est appelé problème de filtrage. Une substitution σ est dite solution du problème $\langle \Gamma, a, b\rangle$ si elle est bien typée dans Γ et les termes σa et σb sont équivalents.*

3 Preuves d'Indécidabilité

On remarque tout d'abord que la preuve d'indécidabilité de l'unification du second ordre dans le λ-calcul simplement typé (Goldfarb [3]) est valide dans tous les calculs du cube. En effet, en reprenant les notations de [3], si H est un système d'équations arithmétiques et S est le problème d'unification codant ce système, comme dans le λ-calcul simplement typé, pour toute solution θ de S, les variables F_i sont instanciées par des termes de la forme:

$$\theta F_i = [w_1 : U]\overline{n_i}\ w_1$$

et les variables G_l pour $l = 2^i 3^j 5^k$ par des termes de la forme:

$$\theta G_l = [w_1 : U][w_2 : U][w_3 : U](g\ (t_0^l\ w_1\ w_2)\ (g\ (t_1^l\ w_1\ w_2)\ ...\ (g\ (t_{n_j-1}^l\ w_1\ w_2)\ w_3)))$$

avec:

$$t_p^l = [w_1 : U][w_2 : U](g\ (\overline{n_i.p}\ w_1)\ (\overline{p}\ w_2))$$
$$\overline{n} = [w_1 : U](g\ a\ ...\ (g\ a\ w_1))\ (n \text{ fois})$$

de tout unificateur de S on peut donc déduire une solution de H.

Proposition 2 (Goldfarb) *Dans un calcul quelconque du cube il n'existe pas de méthode effective permettant de décider si un problème d'unification du second ordre a une solution.*

De plus :

Définition 12 *Un problème d'unification $\langle \Gamma, a, b\rangle$ est dit élémentaire au niveau des termes si $\Gamma = [\forall U : Prop]\Gamma'$, pour toute variable $Qx : T$ déclarée dans Γ', le terme T est l'un des termes U, $U \to U$, $U \to U \to U$ ou $U \to U \to U \to U$ et le type commun de a et b est U.*

Dans un calcul avec constructeurs de types, un problème d'unification est dit élémentaire au niveau des types si pour toute variable $Qx : T$ déclarée dans Γ, le terme T est l'un des termes $Prop$, $Prop \to Prop$, $Prop \to Prop \to Prop$ ou $Prop \to Prop \to Prop \to Prop$ et le type commun de a et b est $Prop$.

Proposition 3 *Dans un calcul quelconque du cube, il n'existe pas de méthode effective permettant de décider si un problème d'unification élémentaire au niveau des termes a une solution.*

Dans un calcul avec constructeurs de type, il n'existe pas de méthode effective permettant de décider si un problème d'unification élémentaire au niveau des types a une solution.

Théorème 1 *Dans un calcul avec types dépendants, il n'existe pas de méthode effective permettant de décider si un problème de filtrage dont toutes les variables existentielles sont du troisième ordre au plus a une solution.*

Preuve. Pour tout problème d'unification élémentaire au niveau des termes $\langle \gamma, u_1, u_2\rangle$ on construit un problème de filtrage $\langle \Gamma, t_1, t_2\rangle$ tel que tous les types des variables existentielles de Γ soient du troisième ordre au plus et $\langle \gamma, u_1, u_2\rangle$ a une solution si et seulement si $\langle \Gamma, t_1, t_2\rangle$ en a aussi une. On pose:

$$\Gamma = \gamma[\forall z : U; \forall P : U \to Prop; \forall c : (P\ z); \forall d : (P\ z); \forall G : (P\ z) \to (P\ z) \to (P\ z);$$

$$\exists f : (h : U \to U)(P\ (h\ u_1)) \to (P\ (h\ u_2))]$$

$$t_1 = (G\ (f\ [x : U]z\ c)\ (f\ [x : U]z\ d)) \qquad t_2 = (G\ c\ d)$$

S'il existe une substitution τ telle que $\tau u_1 = \tau u_2$ alors on pose :

$$\sigma = \tau \cup \{\langle f, [], [x_1 : U \to U][x_2 : (P\ (x_1\ (\tau u_1)))]x_2\rangle\}$$

La substitution σ est bien-typée dans Γ et est solution du problème de filtrage $\sigma t_1 = t_2$.

Réciproquement, si il existe une substitution σ bien-typée dans Γ telle que $\sigma t_1 = t_2$, alors σ est bien-typée dans γ et on montre que $\sigma u_1 = \sigma u_2$. Soit :

$$\Delta = \Gamma[\forall x_1 : U \to U; \forall x_2 : (P\ (x_1\ (\sigma u_1)))]$$

$$v = ((\sigma f)\ x_1\ x_2) : (P\ (x_1\ (\sigma u_2)))$$

L'équation $\sigma t_1 = t_2$ est équivalente au système :

$$v[x_1 \leftarrow [x : U]z, x_2 \leftarrow c] = c \quad v[x_1 \leftarrow [x : U]z, x_2 \leftarrow d] = d$$

Le terme v n'est pas une abstraction ni un produit parce que son type est $(P\ (x_1\ (\sigma u_2)))$. C'est donc un terme atomique $v = (x\ r_1\ ...\ r_p)$ où x est une variable ou l'un des symboles $Prop$ et $Type$. Le symbole x est l'une des variables x_1, x_2, c parce que $v[x_1 \leftarrow [x : U]z, x_2 \leftarrow c] = c$. Il est différent de c parce que $v[x_1 \leftarrow [x : U]z, x_2 \leftarrow d] = d$. Il est différent de x_1 parce que le type de v est différent de $U \rightarrow U$ et U. Donc $x = x_2$. Comme le type de x_2 est atomique $p = 0$ et $v = x_2$. Les termes v et x_2 ont donc même type et on en déduit :

$$\sigma u_1 = \sigma u_2$$

Théorème 2 *Dans un calcul avec constructeurs de types, il n'existe pas de méthode effective permettant de décider si un problème de filtrage dont toutes les variables existentielles sont du troisième ordre au plus a une solution.*

Preuve. Pour tout problème d'unification élémentaire au niveau des types $\langle \gamma, u_1, u_2 \rangle$ on construit un problème de filtrage $\langle \Gamma, t_1, t_2 \rangle$ tel que tous les types des variables existentielles de Γ soient du troisième ordre au plus et $\langle \gamma, u_1, u_2 \rangle$ a une solution si et seulement si $\langle \Gamma, t_1, t_2 \rangle$ en a aussi une. On pose :

$$\Gamma = \gamma[\forall Z : Prop; \forall c : Z; \forall d : Z; \forall G : Z \rightarrow Z \rightarrow Z; \exists f : (h : Prop \rightarrow Prop)(h\ u_1) \rightarrow (h\ u_2)]$$

$$t_1 = (G\ (f\ [X : Prop]Z\ c)\ (f\ [X : Prop]Z\ d)) \qquad t_2 = (G\ c\ d)$$

S'il existe une substitution τ telle que $\tau u_1 = \tau u_2$ alors on pose :

$$\sigma = \tau \cup \{\langle f, [], [x_1 : Prop \rightarrow Prop][x_2 : (x_1\ (\tau u_1))]x_2 \rangle\}$$

La substitution σ est bien-typée dans Γ et est solution du problème de filtrage $\sigma t_1 = t_2$.

Réciproquement, si il existe une substitution σ bien-typée dans Γ telle que $\sigma t_1 = t_2$, alors σ est bien-typée dans γ et on montre que $\sigma u_1 = \sigma u_2$. Soit :

$$\Delta = \Gamma[\forall x_1 : Prop \rightarrow Prop; \forall x_2 : (x_1\ (\sigma u_1))]$$

$$v = ((\sigma f)\ x_1\ x_2) : (x_1\ (\sigma u_2))$$

L'équation $\sigma t_1 = t_2$ est équivalente au système :

$$v[x_1 \leftarrow [X : Prop]Z, x_2 \leftarrow c] = c \quad v[x_1 \leftarrow [X : Prop]Z, x_2 \leftarrow d] = d$$

Le terme v n'est pas une abstraction ni un produit parce que son type est $(x_1\ (\sigma u_2))$. C'est donc un terme atomique $v = (x\ r_1\ ...\ r_p)$ où x est une variable ou l'un des symboles $Prop$ et $Type$. Le symbole x est l'une des variables x_1, x_2, c parce que $v[x_1 \leftarrow [X : Prop]Z, x_2 \leftarrow c] = c$. Il est différent de c parce que $v[x_1 \leftarrow [X : Prop]Z, x_2 \leftarrow d] = d$. Il est différent de x_1 parce que le type de v est différent de $Prop \rightarrow Prop$ et $Prop$. Donc $x = x_2$. Comme le type de x_2 est atomique $p = 0$ et $v = x_2$. Les termes v et x_2 ont donc même type et on en déduit :

$$\sigma u_1 = \sigma u_2$$

Conclusion

Le filtrage est donc indécidable dès le troisième ordre dans les calculs $\lambda P(LF)$, $\lambda P2$, $\lambda\underline{\omega}$, $\lambda\omega(F_\omega)$, $\lambda P\underline{\omega}$ et $\lambda P\omega(CoC)$. Restent ouverts le problème classique de la décidabilité du filtrage dans le λ-calcul simplement typé (problème conjecturé décidable dans [4]) et celui de la décidabilité du filtrage dans le système $\lambda 2(F)$.

References

1. H. Barendregt, Introduction to Generalized Type Systems, To appear in *Journal of Functional Programming.*
2. C. M. Elliott, Higher-order Unification with Dependent Function Types, *Proceedings of the 3^{rd} International Conference on Rewriting Techniques and Applications*, N. Dershowitz (Ed.), Lecture Notes in Computer Science, 355, Springer-Verlag, 1989, pp.121-136.
3. W.D. Goldfarb, The Undecidability of the Second-Order Unification Problem, *Theoretical Computer Science*, 13, 1981, pp. 225-230.
4. G. Huet, Résolution d'Équations dans les Langages d'Ordres 1,2, ..., ω, *Thèse de Doctorat d'État*, Université de Paris VII, 1976.

Auteur: Gilles Dowek, INRIA, Domaine de Voluceau-Rocquencourt, B.P. 105, 78153 Le Chesnay Cedex.

Erratum au compte rendu : L'Indécidabilité du Filtrage du Troisième Ordre dans les Calculs avec Types Dépendants ou Constructeurs de Types

Gilles Dowek

Dans le compte rendu "L'indécidabilité du filtrage du troisième ordre dans les calculs avec types dépendents et constructeurs de types" (I, 312, 12, 1991, pp. 951-956), le second théorème (p. 956) est erroné.

In the note "The undecidability of third order pattern matching in calculi with dependent types or types constructors" (I, 312, 12, 1991, pp. 951-956), the second theorem (p. 956) is invalid.

M. Bezem et J. Springintveld ont trouvé une erreur dans la démonstration du second théorème de ce compte rendu. Le terme $(h : Prop \to Prop)(h\ u_1) \to (h\ u_2)$ n'est bien typé que dans un calcul comprenant des constructeurs de types et des types polymorphes. Ce résultat d'indécidabilité ne s'applique donc pas aux calculs $\lambda\underline{\omega}$ ni $\lambda P\underline{\omega}$. Par ailleurs, l'ordre de ce type n'est pas 3 mais ∞. Le résultat démontré est donc l'indécidabilité du filtrage (et non du filtrage du troisième ordre) dans les calculs avec types polymorphes et constructeurs de types (et non dans les calculs avec constructeurs de types).

M. Bezem et J. Springintveld ont montré qu'il est possible d'adapter la démonstration de façon à obtenir l'indécidabilité du filtrage du quatrième ordre dans les calculs avec types polymorphes et constructeurs de types, en considérant le problème $\langle \Gamma, t_1, t_2 \rangle$ où

$$\Gamma = [\forall P : Prop \to Prop; \forall Z : Prop; \forall c : (P\ Z); \forall d : (P\ Z);$$

$$\forall G : (P\ Z) \to (P\ Z) \to (P\ Z); \exists f : (h : Prop \to Prop)(P\ (h\ u_1)) \to (P\ (h\ u_2))]$$

$$t_1 = (G\ (f\ [X : Prop]Z\ c)\ (f\ [X : Prop]Z\ d))$$

$$t_2 = (G\ c\ d)$$

J. Springintveld a par ailleurs montré la décidabilité du filtrage du troisième ordre dans le système $\lambda\underline{\omega}$, mais l'indécidabilité de ce problème quand on impose aux solutions d'être closes. La décidabilité du filtrage dans le système $\lambda\underline{\omega}$ ainsi que celle du filtrage du troisième ordre dans $\lambda\omega$ restent ouvertes.

Le premier théorème du compte rendu, concernant les calculs avec types dépendants n'est pas affecté.