International Journal of Computational and Experimental Science and Engineering (IJCESEN)
Vol. 1-No.2 (2015) pp. 20-23
http://dergipark.ulakbim.gov.tr/ijcesen

ISSN: **2149-9144**

Research Article

An Optimization of Lambda Type assignments via Resource Control[#]

Jelena IVETIĆ[1]*, Silvia GHILEZAN[1], Nenad SAVIĆ[1]

[1]University of Novi Sad, Faculty of Technical Sciences, Novi Sad - Serbia

* Corresponding Author : jelenaivetic@uns.ac.rs

[#] Presented in "2nd International Conference on Computational and Experimental Science and Engineering (ICCESEN-2015)"

Keywords
Lambda calculus
Type assignment
Resource control
Optimization

Abstract: The size of a lambda term's type assignment is traditionally interpreted as the number of involved typing rules, but can be also assessed using some finer-grained measures such as the number of involved type declarations (TD). We propose a type assignment method that relies on the translation of a typeable lambda term to the corresponding term of a modified resource control lambda calculus. The translation output of a given lambda term is often syntactically more complex, therefore more rules need to be used for its type assignment in the target calculus. However, we show that TD measure decreases when types are assigned to terms satisfying a certain minimal level of complexity, thus our method represents an optimization of the lambda calculus' type assignment.

1. Introduction

The untyped λ-calculus is a simple formal system for expressing all effectively computable functions (equivalent to Turing machine). Typed λ-calculus [1] is a restricted system, where application is controlled by objects (types) assigned to λ-terms. Typed λ-calculus has played an important role in the development of programming languages, proof theory, evaluation strategies, mathematical linguistics, etc... Today, a variety of λ-based calculi and type systems have found application in programming languages for certified compilers, automated theorem provers and proof assistants, software verification, etc...

In the λ-calculus, complexity of computation refers to the number of required reduction rules in order to reach the normal form (if possible). For example, in [4], a particular subclass of λ-terms that reduce to normal form in polynomial time is characterized by means of typeability in a system corresponding to light affine logic. Complexity of type assignment, on the other hand, measures the size of type assignment (TA). It can be assessed via different measures, such as: the number of applied TA rules (length of the proof) - time complexity; the number of used type declarations (weight of the proof) - space complexity; the number of used type variables (also weight of the proof, but not interesting for simple types where the type of a given term is unique). The problem of the weight of the proof reducing is interesting for applicative purposes, since it corresponds to the required memory for the type assignment or type checking. The goal of this paper is to investigate the optimization of the lambda type assignment by using the lambda calculus with resource control, $\lambda_{\circledR}$, proposed in [2].

1 RESOURCE CONTROL IN TYPE ASSIGNMENT OPTIMIZATION

The $\lambda_{\circledR}$-calculus is an extension of the λ-calculus with operators that perform quantitative control of variables by explicitly denoting duplication and erasure of each variable, which on the logical side corresponds to structural rules of weakening and

contraction. The *weakening* operator denotes that the variable x does not appear in the term M, whereas the *contraction* operator denotes that two variables x and y play the same role in M, therefore one may think of them as of one duplicated variable z. A detailed account on the $\lambda_{\circledR}$-calculus can be found in [3].

Each λ-term can be represented in the $\lambda_{\circledR}$-calculus using a mapping $[\ ]_{rc}$. In general, a correspondence between sets of λ-terms and $\lambda_{\circledR}$-terms is "one-to-many". It can be showed that the mapping $[\ ]_{rc}$ is constructed in a way that resource operators are put in positions which are optimal from the TA point of view and their interaction is disabled:

Proposition 1. *For each λ-term M, the corresponding $\lambda_{\circledR}$-term $[M]_{rc}$ is in the γω-normal form.*

Simple types are assigned to $\lambda_{\circledR}$-terms by the TA system $\lambda_{\circledR} \rightarrow$. Its main differences with respect to the system $\lambda \rightarrow$ (of simply-typed λ-calculus) are the following: *(i)* two new TA-rules; *(ii)* minimal form of axiom; *(iii)* context-splitting style for the rule with two premises (where Γ,Δ is disjoint union of two bases Γ and Δ). Systems $\lambda \rightarrow$ and $\lambda_{\circledR} \rightarrow$ are represented in Figure 1.

$\lambda \rightarrow$:

$$\frac{}{\Gamma, x : \alpha \vdash x : \alpha}\ (Ax)$$

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \rightarrow \beta}\ (\rightarrow_{intro})$$

$$\frac{\Gamma \vdash M : \alpha \rightarrow \beta \qquad \Gamma \vdash N : \alpha}{\Gamma \vdash MN : \beta}\ (\rightarrow_{elim})$$

$\lambda_{\circledR} \rightarrow$:

$$\frac{}{x : \alpha \vdash x : \alpha}\ (Ax) \qquad \frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \rightarrow \beta}\ (\rightarrow_I)$$

$$\frac{\Gamma \vdash M : \alpha \rightarrow \beta \quad \Delta \vdash N : \alpha}{\Gamma, \Delta \vdash MN : \beta}\ (\rightarrow_E)$$

$$\frac{\Gamma, x : \alpha, y : \alpha \vdash M : \beta}{\Gamma, z : \alpha \vdash z <^{x}_{y} M : \beta}\ (Cont)$$

$$\frac{\Gamma \vdash M : \alpha}{\Gamma, x : \beta \vdash x \odot M : \alpha}\ (Weak)$$

Figure 1. *Type assignment rules for simply typed λ-calculus and for $\lambda_{\circledR}$-calculus*

It is easy to prove the preservation of the type of a λ-term in the system with resource control:

Proposition 2. *If $\Gamma \vdash M : \alpha$ in $\lambda\rightarrow$, then $\Gamma' \vdash [M]_{rc} : \alpha$ in $\lambda_{\circledR} \rightarrow$, for some $\Gamma' \subseteq \Gamma$.*

Our work plan is as follows. For a given λ-term M:

(i) we translate it to corresponding $\lambda_{\circledR}$-term $[M]_{rc}$;

(ii) proposition 2 ensures the preservation of type;

(iii) due to Proposition 1, we can separately investigate the effects of explicit weakening and contraction on the complexity of TA.

Obviously, the number of TA rules will increase because the system $\lambda_{\circledR} \rightarrow$ consists of more rules which reflects the more complex syntax of the $\lambda_{\circledR}$-term, but what about the number of type declarations?

The effect of explicit weakening is firstly investigated on a simple example of a term with *n* void lambda abstractions ($M \equiv \lambda x_n x_{n-1} \ldots x_1.y$) whose counterpart in the $\lambda_{\circledR}$-calculus contains *n* weakenings ($[M]_{rc} \equiv \lambda x_n.x_n \odot (\lambda x_{n-1}.x_{n-1} \odot (\ldots(\lambda x_1.x_1 \odot y)\ldots))$). The type $\alpha_n \rightarrow \alpha_{n-1} \rightarrow \cdots \rightarrow \alpha_1 \rightarrow \beta$ can be assigned to both M and $[M]_{rc}$, but the complexity of TA differs. In the system $\lambda\rightarrow$, the number of applied TA rules is $n + 1$ and the number of used TD is $(n + 1)(n + 2)/2$. On the other hand, in the system $\lambda_{\circledR} \rightarrow$, the number of applied TA rules is $2n + 1$ but the number of used TD is $3n + 1$. Therefore, despite of the increased number of applied TA rules, the total number of used TD is smaller in the presence of explicit weakening for all terms with $n > 3$.

Even in the more general case where *n* void abstractions (i.e. weakenings) are distributed along the term, we obtain that in the system $\lambda \rightarrow$ the number of TD behaves as $O(n^2)$, whereas in the system $\lambda_{\circledR} \rightarrow$ number of TD behaves as $O(n)$.

Next, in order to explore the effect of explicit contraction, we firstly analyze an example of the λ-term with *n* repetitions of the same variable. This is Church numeral representing the natural number N: $N \equiv \lambda f.\lambda x.f^n(x)$. The type int: $(\alpha \rightarrow \alpha) \rightarrow \alpha \rightarrow \alpha$ can be assigned to both N and its resource sensitive counterpart $[N]_{rc}$ which contains *n-1* contractions. In the system $\lambda \rightarrow$, the number of applied TA rules is $2n + 3$; the number of TD: $4n + 3$. In the system $\lambda_{\circledR} \rightarrow$, the number of TA rules is $3n + 2$; the number of TD: $6n - 1$. In both cases the number of TD behaves as $O(n)$, but without explicit contraction it is smaller for $n \geq 2$. Therefore, in spite of the presence of the context splitting rule for application which decreases number of TD, explicit contraction in general does not contribute to the optimization of the type assignment complexity. An explanation for

this phenomenon lies in the fact that apart from the additional TA rules, the number of TD is enlarged because all fresh variables need to be declared. This neutralizes the positive effect of the context-splitting rule.

Conclusion of analysis of effects: In case of the explicit weakening, it is not possible to separate two additional features of the system $\lambda_{\circledR} \rightarrow$: the minimal axiom and the new rule *(Weak)*. But, in case of the explicit contraction, it is possible to separate two additional features of the system: context splitting form of the rule *(→E)* and the new rule *(Cont)*. Thus, we can build a novel, *hybrid system* $\lambda^o\rightarrow$, collecting only those features that reduce the number of TD:

(i) explicit weakening;
(ii) implicit contraction;
(iii) partial context-splitting form of the rule *(→E)* (with ordinary union $\Gamma \cup \Delta$ instead of disjoint one Γ,Δ).

Syntax

Pre-terms:

$$M, N \quad ::= \quad x \mid \lambda x.M \mid MN \mid x \odot M$$

Terms are pre-terms that satisfy the condition:

in

$$\lambda x.M \quad x \in Fv(M)$$

Reductions

$$\begin{array}{lrcl} (\beta) & (\lambda x.M)N & \rightarrow & M[N/x] \\ (\omega_1) & \lambda x.(y \odot M) & \rightarrow & y \odot (\lambda x.M) \\ (\omega_2) & (x \odot M)N & \rightarrow & x \odot (MN) \\ (\omega_3) & M(x \odot N) & \rightarrow & x \odot (MN) \end{array}$$

Mapping from λ

$$\begin{array}{rcl} [x]_o & = & x \\ [MN]_o & = & [M]_o[N]_o \\ [\lambda x.M]_o & = & \begin{cases} \lambda x.[M]_o, & x \in Fv(M) \\ \lambda x.x \odot [M]_o, & x \notin Fv(M) \end{cases} \end{array}$$

System λ°→

$$\frac{}{x : \alpha \vdash x : \alpha}\ (Ax)$$

$$\frac{\Gamma, x : \alpha \vdash M : \beta}{\Gamma \vdash \lambda x.M : \alpha \rightarrow \beta}\ (\rightarrow_I)$$

$$\frac{\Gamma \vdash M : \alpha \rightarrow \beta \quad \Delta \vdash N : \alpha}{\Gamma \cup \Delta \vdash MN : \beta}\ (\rightarrow_E)$$

$$\frac{\Gamma \vdash M : \alpha}{\Gamma, x : \beta \vdash x \odot M : \alpha}\ (Weak)$$

***Figure** 2. The λ° – calculus*

The syntax and reduction rules of the λ^o-calculus, the mapping $[\ ]_o$ from λ to λ^o , and type assignment system $\lambda^o\rightarrow$ are given in Figure 2.

Turning back to Church numeral N and the type assignment of its corresponding term $[N]^o$ in the system $\lambda^o\rightarrow$, we obtain the following results: the number of TA rules is $2n + 3$; the number of TD: $3n + 2$ (which is less than in $\lambda\rightarrow$ for all n). Finally, some additional results for several randomly chosen terms, typed in parallel in the systems $\lambda\rightarrow$ and $\lambda^o\rightarrow$ are given in Table 1. Since some terms are too long to fit in the table, we will give their standard interpretation instead of the syntax. For example, we will write 3 for $\lambda f.\lambda x.f(f(fx))$, MULT for $\lambda x.\lambda y.\lambda z.x(yz)$, FALSE for $\lambda x.\lambda y.y$, etc… We will also use prefix notation for binary operations. For example, PLUS 10 (MULT 5 2) stands for $10 + 5*2$.

***Table** 1. Some empirical results*

(meaning of) the term	Number of TD in $\lambda\rightarrow$	Number of TD in $\lambda^o\rightarrow$	% of improvement
MULT 3 2	44	30	32%
PLUS 10 (MULT 5 2)	137	90	34%
AND TRUE FALSE	17	14	18%
λw.(λx.yz)(uv)	47	26	45%

2 CONCLUSION

The λ^o-calculus is a modified version of the $\lambda_{\circledR}$-calculus, which preserves good properties of the system $\lambda_{\circledR} \rightarrow$ relevant for the optimization issues. Except for the limited number of simple terms, it reduces the type assignment procedure in terms of the number of type declarations, thus optimizing required space (= memory) for type checking, which makes it suitable for applications. To review the full extent of the proposed optimization method, we will in the future conduct semi-automated testing with large-scale terms. Also, it would be interesting to extend the proposed method to more complex type-systems (such as intersection types, polymorphic types, dependent types,...) and to other lambda-based formal calculi.

REFERENCES

[1] H. P. Barendregt, W. Dekkers, and R. Statman. Lambda Calculus with Types. Perspectives in logic. Cambridge University Press, 2013.

[2] S. Ghilezan, J. Ivetić, P. Lescanne, and S. Likavec. Intersection types for the resource control lambda calculi. In A. Cerone and P. Pihlajasaari, editors, 8th International Colloquium on Theoretical Aspects of Computing, ICTAC '11, volume 6916 of Lecture Notes in Computer Science, pages 116–134. Springer, 2011. *DOI: 10.1007/978-3-642-23283-1_10*

[3] S. Ghilezan, J. Ivetić, P. Lescanne, and S. Likavec. Resource control and intersection types: an intrinsic connection. CoRR, abs/1412.2219, 2014.

[4] P. Baillot, K. Terui. Light types for polynomial time computation in lambda calculus. Information and Computation, volume 207, pages 41-62. Elsevier, 2009. *DOI:10.1016/j.ic.2008.08.005*