A compact encoding for λ-terms in interaction calculus

Anton Salikhmetov

May 1, 2014

Abstract

An extremely compact encoding for λ-terms straight into the calculus for interaction nets is introduced. We achieve balance between the minimal set of symbols, the minimal net to represent a term, and reasonable efficiency. Also, we construct a directed version of the system.

1 Introduction

There are two extreme ways to encode λ-terms [1] into interaction nets [2]. They pursue conflicting goals. The original optimal one by Lamping [3] implements β-reduction of λ-terms the asymptotically most efficient way. The other one [4] represents λ-terms in an interaction system with the minimal set of symbols, namely the set of interaction combinators [5]. (In fact, there exists a universal interaction system by Bechet [6] with only two agents instead of three, but its interaction rules appear to be too complicated.) However, both encodings have their own downsides. The set of symbols used by Lamping’s encoding is practically infinite due to the use of integer labels for agents. The encoding in interaction combinators in turn appears not to be optimal by Levy. Besides, both encodings involve some overhead to represent a λ-term.

We present a compact representation of λ-terms pursuing balance between the minimal set of symbols and efficiency of implementation. To define our encoding in a more concise and formal way, we will use the calculus for interaction nets [7] rather than graphical representation. Namely, λ-terms will be mapped into configurations. Also, we construct a directed version of the resulting interaction system similar to that of directed combinators [5].

2 Encoding λ-terms

We work in interaction calculus [7].

Let λ, ψ, δ, ε ∈ Σ, Ar(λ) = Ar(ψ) = Ar(δ) = 2, and Ar(ε) = 0.
For any $\alpha \in \Sigma$, $\beta \in \{\psi, \delta\}$, $\alpha \neq \beta$, and $\text{Ar}(\alpha) = n$, we assume
\[
\begin{align*}
\alpha[x_1, \ldots, x_n] & \Rightarrow \alpha[x_1, \ldots, x_n]; \\
\alpha[\delta(x_1, y_1), \ldots, \delta(x_n, y_n)] & \Rightarrow \beta[\alpha(x_1, \ldots, x_n), \alpha(y_1, \ldots, y_n)]; \\
\alpha[\epsilon, \ldots, \epsilon] & \Rightarrow \epsilon.
\end{align*}
\]

Any $\lambda$-term $M$ can be mapped into a configuration $\langle x \mid \Gamma(M, x) \rangle$ as follows:
\[
\begin{align*}
\Gamma(y, x) &= \{x = y\}; \\
\Gamma(\lambda y. M, x) &= \{x = \lambda(\epsilon, z)\} \cup \Gamma(M, z), \quad \text{when } y \not\in \text{FV}(M); \\
\Gamma(\lambda y. M, x) &= \{x = \lambda(y, z)\} \cup \Gamma(M, z), \quad \text{when } y \in \text{FV}(M); \\
\Gamma(M \ N, x) &= \{y = \lambda(z, x), t_1 = \psi(t'_1, t''_1), \ldots, t_n = \psi(t'_n, t''_n)\} \cup \Gamma(M', y) \cup \Gamma(N', z),
\end{align*}
\]
where
\[
\{t_1, \ldots, t_n\} = \text{FV}(M) \cap \text{FV}(N); \\
M' = M[t_1 := t'_1] \ldots [t_n := t'_n]; \\
N' = N[t_1 := t''_1] \ldots [t_n := t''_n].
\]

3 A directed version of the encoding

The above encoding has balance between the minimal set of symbols, the minimal net to represent a term, and efficiency of the reduction implementation. It has completely no overhead to represent applications, abstractions, and sharing. Besides, we can see that the interaction rules have unified form similar to those for directed combinators [5].

Directed combinators have some advantages over the usual interaction combinators from the viewpoint of software implementation. Specifically, an interaction rule is easier to choose in that system, because the active pairs of directed combinators can be represented as ordered pairs, unlike unordered active pairs in usual interaction systems.

In [5], interaction combinators are represented using directed combinators with two directed combinators per each interaction combinator. However, the directed version of our interaction system can be constructed without doubling the agents to encode $\lambda$-terms. Instead, we use a different technique to preserve polarity between the ports of agents. Namely, let $\Sigma = \{\lambda, \lambda^*, \delta, \delta^*, \psi, \psi^*\}$, $\forall \alpha \in \Sigma : \text{Ar}(\alpha) = 2$, and the set $\mathcal{R}$ of interaction rules have the following nine elements:
\[
\begin{align*}
\lambda[a, b] & \Rightarrow \lambda^*[a, b], \quad \delta[a, b] \Rightarrow \delta^*[a, b], \quad \psi[a, b] \Rightarrow \psi^*[a, b]; \\
\lambda[\delta(a, b), \delta^*(c, d)] & \Rightarrow \delta^*[\lambda(a, c), \lambda(b, d)], \quad \lambda[\delta(a, b), \delta^*(c, d)] \Rightarrow \psi^*[\lambda(a, c), \lambda(b, d)]; \\
\delta[\lambda^*(a, b), \lambda^*(c, d)] & \Rightarrow \lambda^*[\delta^*(a, c), \delta(b, d)], \quad \psi[\lambda^*(a, b), \lambda^*(c, d)] \Rightarrow \lambda^*[\delta^*(a, c), \delta(b, d)]; \\
\psi[\delta^*(a, b), \delta^*(c, d)] & \Rightarrow \delta^*[\psi(a, c), \psi(b, d)], \quad \delta[\psi^*(a, b), \psi^*(c, d)] \Rightarrow \psi^*[\delta(a, c), \delta(b, d)].
\end{align*}
\]

We omit the erasing agent $\epsilon$ which only performs garbage collection in disconnected nets.

One can notice duality between the agents $\lambda, \delta, \psi$ and $\lambda^*, \delta^*, \psi^*$, respectively. That is why we produce two dual encodings for an arbitrary $\lambda$-term, namely configurations
\langle x | \Gamma(M, x) \rangle \text{ and } \langle x | \Gamma^*(M, x) \rangle, \text{ where the } \Gamma \text{ and } \Gamma^* \text{ mappings are defined as follows:}

\Gamma(y, x) = \Gamma^*(y, x) = \{x = y\};
\Gamma(\lambda y. M, x) = \{x = \lambda(y, z)\} \cup \Gamma(M, z);
\Gamma^*(\lambda y. M, x) = \{x = \lambda^*(y, z)\} \cup \Gamma^*(M, z);
\Gamma(M \ N, x) = \{y = \lambda(z, x), t_1 = \psi^*(t_1', t_2', \ldots, t_n = \psi^*(t_n', t_n'')\} \cup \Gamma(M', y) \cup \Gamma(N', z),
\Gamma^*(M \ N, x) = \{y = \lambda(z, x), t_1 = \psi(t_1', t_2'), \ldots, t_n = \psi(t_n', t_n'')\} \cup \Gamma^*(M', y) \cup \Gamma^*(N', z),

\text{where}

\{t_1, \ldots, t_n\} = \text{FV}(M) \cap \text{FV}(N);
M' = M[t_1 := t_1'] \ldots [t_n := t_n'];
N' = N[t_1 := t_1''] \ldots [t_n := t_n''].

These two dual encodings were found through the following observation: when using unordered representation of \(\lambda\)-terms, the orientation of all application and sharing agents with respect to abstraction agents is preserved during evaluation.

References

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

[2] Y. Lafont. Interaction nets. Proceedings, 17th ACM Symposium on Principles of Programming Languages, 95–108. ACM Press, 1990.

[3] J. Lamping. An algorithm for optimal lambda calculus reduction. 17th Annual Symposium on Principles of Programming Languages, 16–30. ACM Press, 1990.

[4] I. Mackie, J. S. Pinto. Compiling the Lambda Calculus into Interaction Combinators. Logical Abstract Machines workshop, 1998.

[5] Y. Lafont. Interaction combinators. Information and Computation, 137(1): 69–101, 1997.

[6] D. Bechet. Universal interaction systems with only two agents. 12th International Conference on Rewriting Techniques and Applications. Utrecht, The Netherlands, 2001.

[7] M. Fernández, I. Mackie. A calculus for interaction nets. Proceedings of the International Conference on Principles and Practice of Declarative Programming, vol. 1702 of Lecture Notes in Computer Science, 170–187. Springer-Verlag, 1999.

[8] M. Fernández, I. Mackie. Operational equivalence for interaction nets. Theoretical Computer Science, vol. 297, 157–181.