SECD Machine Implementation of $\lambda_{\rightarrow}$

TAZMILUR SAAD, Independent, USA

This paper presents an OCaml implementation of the simply typed lambda calculus extended with units, sums and products using the SECD machine.

1 INTRODUCTION

This work documents an autodidactic implementation of an interpreter for a simply typed lambda calculus extended unit, sums and product types. The interpreter is based on the SECD machine.

2 CALCULUS

Contexts are an ordered list of variables equipped with a type.

$$\Gamma \coloneqq \varnothing \mid \Gamma, x : \tau$$

The types in our system includes the unit type, sum types, product types, function types and the base types.

$$\tau \coloneqq \mathbb{1} \mid \tau + \tau \mid \tau \times \tau \mid \tau \rightarrow \tau \mid \texttt{Nat} \mid \texttt{Bool}$$

The syntax of the system includes variables, abstractions, applications, let bindings, if-then-else and members of the base types of natural numbers and booleans.

$$e \coloneqq x \mid \lambda x : \tau . e \mid e_1 e_2 \mid \text{let } x = e_1 \text{ in } e_2 \mid \text{ if } e_1 \text{ then } e_2 \text{ else } e_3 \mid \mathbb{N} \mid \mathbb{B}$$

The typing judgements and the operational semantics are shown in Figure 1 and 2 respectively.

3 SOUNDNESS

Theorem 3.1 (Progress). *If $\cdot \vdash x : \tau$ is a well typed term then x is a value or there exists some y such that $x \mapsto y$.*

Proof. Admitted. □

Theorem 3.2 (Preservation). *If $\cdot \vdash x : \tau$ and $x \mapsto y$, then $\cdot \vdash y : \tau$.*

Proof. Admitted. □

4 IMPLEMENTATION

Typing contexts are implemented using a locally nameless representation.

5 CONCLUSION

We have implemented an interpreter for the simply typed lambda calculus.

6 ACKNOWLEDGEMENTS

Thanks to Anton Lorenzen for his guidance on the project.

Author's address: Tazmilur Saad, Independent, Jersey City, NJ, USA, ssaad@colgate.edu.

$$\frac{}{\Gamma \vdash () : \mathbb{1}}\ \textsc{Unit} \qquad \frac{}{\Gamma \vdash \mathbb{N} : \texttt{Nat}}\ \textsc{Nat} \qquad \frac{}{\Gamma \vdash \mathbb{B} : \texttt{Bool}}\ \textsc{Bool}$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ \textsc{Var} \qquad \frac{\Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 . e_1 \tau_1 \to \tau_2}\ \textsc{Lam} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash e_1 e_2 : \tau_2}\ \textsc{App}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \text{ let } x = e_1 \text{ in } e_2 : \tau_2}\ \textsc{Let} \qquad \frac{\Gamma \vdash e_1 : \texttt{Bool} \qquad \Gamma \vdash e_2 : \tau \qquad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \text{ if } e_1 \text{ then } e_2 \text{ else } e_3 : \tau}\ \textsc{Cond}$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \text{ inl } e : \tau_1 + \tau_2}\ \textsc{Inl} \qquad \frac{\vdash e : \tau_2}{\Gamma \vdash \text{ inr } e : \tau_1 + \tau_2}\ \textsc{Inr}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}\ \textsc{Pair} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \times \tau_2 \qquad \Gamma, x : \tau_1, y : \tau_2 \vdash e_2 : \tau_3}{\Gamma \vdash \text{ let } (x, y) = e_1 \text{ in } e_2 : \tau_3}\ \textsc{Split}$$

Fig. 1. Simply typed lambda calculus

Fig. 2. Operational Semantics