Type Theory 2011 – Parigot's λ_μ-calculus

Robbert Krebbers

November 1, 2011

1 Terms and types

Whereas most other systems with control, for example $\lambda_{\mathcal{C}}$ [FF86, Gri90] or λ_Δ [RS94], use ordinary λ-variables to represent continuations, λ_μ distinguishes λ-variables from continuation variables. Also, the terms are of a more restricted shape because the system distinguishes between *terms* and *commands*.

Definition 1.1. Simple types *are inductively defined over an infinite set of type variables* $(\alpha, \beta, \ldots)$ *as follows.*

$$\rho, \sigma ::= \alpha \mid \rho \to \sigma$$

An environment $(\Gamma, \Sigma, \ldots)$ *is an association list of types indexed by variables.*

Definition 1.2. Terms *and* commands *of* λ_μ *are mutually inductively defined over an infinite set of* λ-variables $(x, y, \ldots)$ *and* μ-variables $(\alpha, \beta, \ldots)$ *as follows.*

$$\begin{aligned} t, r, s &::= x \mid \lambda x : \rho . r \mid ts \mid \mu\alpha : \rho . c \\ c, d &::= [\alpha]t \end{aligned}$$

Remark 1.3. *The precedence of* $[\alpha]t$ *is weaker than* sr*, so instead of* $[\alpha](sr)$*, we write* $[\alpha]sr$*.*

As usual, we let $\mathrm{FV}(t)$ and $\mathrm{FCV}(t)$ denote the set of free λ-variables and μ-variables of a term t, respectively. Moreover, $t[x := r]$ denotes substitution of r for x in t, which is capture avoiding for both λ- and μ-variables.

Convention 1.4. *Although a λ-abstraction is annotated by a type, we omit these type annotations when they are obvious or not relevant. Furthermore, we use the* Barendregt convention*. That is, given an expression, we may assume that bound variables are distinct from free variables and that all bound variables are distinct.*

Definition 1.5. *The* typing judgment for terms $\Gamma; \Delta \vdash t : \rho$ *and the* typing judgment for commands $\Gamma; \Delta \vdash c : \perp\!\!\!\perp$ *are as shown in Figure 1.*

A typing judgment $\Gamma; \Delta \vdash t : \rho$ is *derivable in* λ_μ in case it is the conclusion of a derivation tree that uses the derivation rules of Definition 1.5. We say that "the term t has type ρ in the environment of λ-variables Γ and the environment of μ-variables Δ".

$$\frac{x : \rho \in \Gamma}{\Gamma; \Delta \vdash x : \rho}$$
(a) axiom

$$\frac{\Gamma, x : \rho; \Delta \vdash t : \sigma}{\Gamma; \Delta \vdash \lambda x : \rho.t : \rho \to \sigma}$$
(b) lambda

$$\frac{\Gamma; \Delta \vdash t : \rho \to \sigma \qquad \Gamma; \Delta \vdash s : \rho}{\Gamma; \Delta \vdash ts : \sigma}$$
(c) app

$$\frac{\Gamma; \Delta, \alpha : \rho \vdash c : \perp\!\!\!\perp}{\Gamma; \Delta \vdash \mu\alpha : \rho.c : \rho}$$
(d) activate

$$\frac{\Gamma; \Delta \vdash t : \rho \qquad \alpha : \rho \in \Delta}{\Gamma; \Delta \vdash [\alpha]t : \perp\!\!\!\perp}$$
(e) passivate

Figure 1: The typing rules of λ_μ.

Similarly, a typing judgment $\Gamma; \Delta \vdash c : \perp\!\!\!\perp$ is *derivable in* λ_μ in case it is the conclusion of a derivation tree that uses the derivation rules of Definition 1.5. We say that "the command c is typable in the environment of λ-variables Γ and the environment of μ-variables Δ".

Since the passivate and activate rule should always be applied consecutively, it is sometimes convenient to combine these rules into one rule.

$$\frac{\Gamma; \Delta, \alpha : \rho \vdash t : \sigma \qquad \beta : \sigma \in (\Delta, \alpha : \rho)}{\Gamma; \Delta \vdash \mu\alpha : \rho.[\beta]t : \rho}$$

2 The Curry-Howard correspondence

The typing rules of λ_μ correspond to those of *free deduction* [Par92]. This paper does not present free logic but instead considers the relation between λ_μ and minimal classical logic. Minimal classical logic is minimal first-order propositional logic with Peirce's law.

$$\frac{\Gamma \vdash (A \to B) \to A}{\Gamma \vdash A}$$

One direction of this correspondence is straightforward.

Lemma 2.1. *If* $\Gamma \vdash A$ *in minimal classical logic, then there is a term* t *such that* $\Gamma; \emptyset \vdash t : A$.

Proof. Implication introduction corresponds to a λ-abstraction and implication elimination corresponds to an application. This leaves us to prove that Peirce's law is typable in λ_μ. Assume that we have a term t of type $(\rho \to \sigma) \to \rho$, now we construct a term of type ρ as follows.

$$\dfrac{\dfrac{\dfrac{\Gamma; \Delta, \alpha : \rho \vdash t\ (\lambda x.\mu\beta.[\alpha]x) : \rho}{\Gamma; \Delta, \alpha : \rho \vdash [\alpha]t\ (\lambda x.\mu\beta.[\alpha]x) : \perp\!\!\!\perp}}{\Gamma; \Delta \vdash \mu\alpha.[\alpha]t\ (\lambda x.\mu\beta.[\alpha]x) : \rho}}{}$$

where the first conclusion is obtained from

$$\frac{\Gamma, x : \rho; \Delta, \alpha : \rho, \beta : \sigma \vdash t : (\rho \to \sigma) \to \rho \qquad \dfrac{\dfrac{\dfrac{\Gamma, x : \rho; \Delta, \alpha : \rho, \beta : \sigma \vdash x : \rho}{\Gamma, x : \rho; \Delta, \alpha : \rho, \beta : \sigma \vdash [\alpha]x : \perp\!\!\!\perp}}{\Gamma, x : \rho; \Delta, \alpha : \rho \vdash \mu\beta.[\alpha]x : \sigma}}{\Gamma; \Delta, \alpha : \rho \vdash \lambda x.\mu\beta.[\alpha]x : \rho \to \sigma}}{\Gamma; \Delta, \alpha : \rho \vdash t\ (\lambda x.\mu\beta.[\alpha]x) : \rho}$$

□

One should think of the proof term $\mu\alpha : \rho.[\alpha]t(\lambda x : \rho.\mu\beta : \sigma.[\alpha]x)$ as follows. Our goal is ρ, which we label α. Since we have $(\rho \to \sigma) \to \rho$ by assumption, it suffices to prove $\rho \to \sigma$. Therefore, let us assume ρ, which we label x. Now our goal is σ, which we label β. However, instead of proving goal β we prove an earlier goal, namely α, which simply follows from the assumption x.

The converse of Lemma 2.1 is a bit harder, because λ_μ has two environments whereas minimal classical logic has just one. This means that both environments have to be merged into a single environment. If we have $\Gamma; \Delta \vdash t : \rho$ in λ_μ, then we certainly have $\Gamma, \neg\Delta \vdash \rho$ in classical logic, because activate corresponds to Reduction Ad Absurdum and passivate to negation elimination. However, this approach fails to work for proving a correspondence with minimal classical logic, because negation cannot be expressed there. To this end, we define a suitable translation of the environment of μ-variables.

Definition 2.2. *Given a term t and a μ-variable β, a set of simple types t_β is defined as follows.*

$$\begin{aligned} x_\beta &:= \emptyset \\ (\lambda x.t)_\beta &:= t_\beta \\ (ts)_\beta &:= t_\beta \cup s_\beta \\ (\mu\alpha : \rho.[\gamma]t)_\beta &:= t_\beta \qquad \textit{provided that } \beta \neq \gamma \\ (\mu\alpha : \rho.[\beta]t)_\beta &:= \{\rho\} \cup t_\beta \end{aligned}$$

Moreover, given a term t and an environment of μ-variables Δ, a set of simple types t_Δ is defined as $t_\Delta := \{\sigma \to \tau \mid \tau \in t_\beta, \beta : \sigma \in \Delta\}$.

Lemma 2.3. *If $\Gamma; \Delta \vdash t : \rho$ in λ_μ, then $\Gamma, t_\Delta \vdash \rho$ in minimal classical logic.*

Proof. By induction on the derivation $\Gamma; \Delta \vdash t : \rho$. The only interesting case is activate/passivate, so let $\Gamma; \Delta \vdash \mu\alpha.[\gamma]t : \rho$ with $\Gamma; \Delta, \alpha : \rho \vdash t : \sigma$ and $\gamma : \sigma \in (\Delta, \alpha : \rho)$. Now we have $\Gamma, t_{(\Delta,\alpha:\rho)} \vdash \sigma$ by the induction hypothesis. Furthermore

$$\begin{aligned} t_{(\Delta,\alpha:\rho)} &= t_\Delta \cup \{\rho \to \tau \mid \tau \in t_\alpha\} \\ &= t_\Delta \cup \{\rho \to \tau_1, \ldots, \rho \to \tau_n\} \end{aligned}$$

for some simple types $\tau_1, \ldots, \tau_n$. Now, by using Peirce's law and implication introduction n times, we have:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\Gamma, (\mu\alpha.[\gamma]t)_\Delta, \rho \to \tau_1, \ldots, \rho \to \tau_n \;\vdash\; \rho}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta, \rho \to \tau_1, \ldots, \rho \to \tau_{n-1} \;\vdash\; (\rho \to \tau_n) \to \rho}}{\ldots \;\vdash\; \ldots}}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta, \rho \to \tau_1 \;\vdash\; \rho}}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta \;\vdash\; (\rho \to \tau_1) \to \rho}}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta \;\vdash\; \rho}$$

We distinguish the cases $\alpha = \gamma$ and $\alpha \neq \gamma$. In the first case we also have $\sigma = \rho$ since $\alpha : \sigma \in (\Delta, \alpha : \rho)$, and hence $\Gamma, t_\Delta, \rho \to \tau_1, \ldots, \rho \to \tau_n \vdash \rho$ using the induction hypothesis. Moreover we have $(\mu\alpha.[\alpha]t)_\Delta = t_\Delta$ because $\alpha \notin \mathrm{dom}(\Delta)$, so by the above derivation we are done.

In the second case we have $(\mu\alpha.[\gamma]t)_\Delta = t_\Delta \cup \{\sigma \to \rho\}$, so by thinning and implication elimination we have:

$$\dfrac{\dots \ \vdash \ \sigma \to \rho \qquad \dfrac{\Gamma, t_\Delta, \rho \to \tau_1, \dots, \rho \to \tau_n \ \vdash \ \sigma}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta, \rho \to \tau_1, \dots, \rho \to \tau_n \ \vdash \ \sigma}}{\Gamma, (\mu\alpha.[\gamma]t)_\Delta, \rho \to \tau_1, \dots, \rho \to \tau_n \ \vdash \ \rho} \qquad \Box$$

Corollary 2.4. *If* $\Gamma; \emptyset \vdash t : \rho$ *in* λ_μ, *then* $\Gamma \vdash \rho$ *in minimal classical logic.*

Proof. By Lemma 2.3 using the fact that $t_\emptyset = \emptyset$. $\Box$

3 Reduction

In order to present the reduction rules we need to define an extra notion of substitution: *structural substitution.* Performing structural substitution of a μ-variable β and a context E for a μ-variable α, notation $t[\alpha := \beta E]$, will recursively replace each command $[\alpha]t$ by $[\beta]E[t']$, where $t' \equiv t[\alpha := \beta E]$.

Definition 3.1. *A* λ_μ-context *is defined as follows.*

$$E ::= \Box \mid Et$$

Definition 3.2. *Given a* λ_μ*-context* E *and a term* s*,* substitution of s for the hole in E*, notation* $E[s]$*, is defined as follows.*

$$\begin{aligned} \Box[s] &:= s \\ (Et)[s] &:= E[s]t \end{aligned}$$

Definition 3.3. Structural substitution $t[\alpha := \beta E]$ *of a* μ*-variable* β *and a* $\lambda\mu$*-context* E *for a* μ*-variable* α *is defined as follows.*

$$\begin{aligned} x[\alpha := \beta E] &:= x \\ (\lambda x.r)[\alpha := \beta E] &:= \lambda x.r[\alpha := \beta E] \\ (ts)[\alpha := \beta E] &:= t[\alpha := \beta E]s[\alpha := \beta E] \\ (\mu\gamma.c)[\alpha := \beta E] &:= \mu\gamma.c[\alpha := \beta E] \\ ([\alpha]t)[\alpha := \beta E] &:= [\beta]E[t[\alpha := \beta E]] \\ ([\gamma]t)[\alpha := \beta E] &:= [\gamma]t[\alpha := \beta E] \qquad \textit{provided that } \gamma \neq \alpha \end{aligned}$$

Structural substitution is capture avoiding for both λ*- and* μ*-variables.*

Example 3.4. *Consider the following examples.*

1. $([\alpha]x\ (\mu\beta.[\alpha]r))[\alpha := \alpha\ (\Box\ s\ t)] \equiv [\alpha]x\ (\mu\beta.[\alpha]r\ s\ t)\ s\ t$
2. $([\alpha]\lambda x.\mu\beta.[\alpha]x)[\alpha := \gamma\ (\Box\ x)] \equiv [\gamma](\lambda z.\mu\beta.[\gamma]z\ x)\ x$

This paper uses a notion of structural substitution that is more general than Parigot's original presentation [Par92]. In Parigot's original presentation one has $t[\beta := \alpha]$, which renames each μ-variable β into α, and $t[\alpha := s]$, which replaces each command $[\alpha]t$ by $[\alpha]t's$, where $t' \equiv t[\alpha := s]$. Of course, Parigot's notions are just instances of our definition, namely, the former corresponds to $t[\beta := \alpha\ \Box]$ and the latter to $t[\alpha := \alpha\ (\Box s)]$. Although Parigot's presentation suffices for the definition of his reduction rules, our presentation turns out to be better suited for extensions and proofs. For example, Geuvers, Krebbers and McKinna [GKM11] use it in a presentation of λ_μ with naturals numbers and primitive recursion for which they prove meta theoretical properties as confluence for untyped terms and strong normalization.

Definition 3.5. *Reduction $t \to t'$ on λ_μ-terms t and t' is defined as the compatible closure of the following rules.*

$$\begin{array}{lll} (\lambda x.t)r & \to_\beta & t[x := r] \\ (\mu\alpha.c)s & \to_{\mu R} & \mu\alpha.c[\alpha := \alpha\ (\Box s)] \\ \mu\alpha.[\alpha]t & \to_{\mu\eta} & t \qquad \textit{provided that } \alpha \notin \mathrm{FCV}(t) \\ [\alpha]\mu\beta.c & \to_{\mu i} & c[\beta := \alpha\ \Box] \end{array}$$

As usual, $\twoheadrightarrow^+$ denotes the transitive closure, $\twoheadrightarrow$ denotes the reflexive/transitive closure and $=$ denotes the reflexive/symmetric/transitive closure.

From a computational point of view one should think of $\mu\alpha.[\beta]t$ as a combined catch and throw clause: it catches exceptions labeled α in t and finally throws the results of t to $\mu\beta.c$.

Notation 3.6. $\Theta c := \mu\gamma : \rho.c$ *provided that* $\gamma \notin \mathrm{FCV}(c)$.

Definition 3.7. *The terms* `catch` $\alpha\ t$ *and* `throw` $\beta\ s$ *are defined as follows.*

$$\begin{aligned} \texttt{catch}\ \alpha\ t &:= \mu\alpha.[\alpha]t \\ \texttt{throw}\ \beta\ s &:= \Theta[\beta]s \end{aligned}$$

Lemma 3.8. *We have the following reductions for* `catch` *and* `throw`*.*

1. $E[\texttt{throw}\ \alpha\ t] \twoheadrightarrow \texttt{throw}\ \alpha\ t$
2. $\texttt{catch}\ \alpha\ (\texttt{throw}\ \alpha\ t) \twoheadrightarrow \texttt{catch}\ \alpha\ t$
3. $\texttt{catch}\ \alpha\ t \twoheadrightarrow t$ *provided that* $\alpha \notin \mathrm{FCV}(t)$
4. $\texttt{throw}\ \beta\ (\texttt{throw}\ \alpha\ s) \to \texttt{throw}\ \alpha\ s$

Proof. These reductions follow directly from the reduction rules of λ_μ, except for the first one, where an induction on the structure of E is needed. □

Notice that our notion of `catch` and `throw` is not the same as `try` and `raise` in OCaml or `catch` and `throw` in Lisp. In those languages exceptions are *dynamically bound*, which means that substitution is not capture avoiding for exception names, while ours are *statically bound*.

Example 3.9. *Consider the following term:*

$$\texttt{catch}\ \alpha\ \mathtt{S}((\lambda f : \mathtt{N} \to \mathtt{N}.\,\texttt{catch}\ \alpha\ (f\,0))\ \lambda x : \mathtt{N}.\,\texttt{throw}\ \alpha\ x).$$

Here, both occurrences of `catch` *bind different occurrences α. So after two β-reduction steps we obtain* $\texttt{catch}\ \alpha\ \mathtt{S}(\texttt{catch}\ \beta\ (\texttt{throw}\ \alpha\ 0))$ *and hence its normal form is* 0*. In systems with dynamically bound exceptions this term would reduce to* $\mathtt{S}0$ *because the* `throw` *would get caught by the innermost* `catch`*.*

4 Some meta theoretical properties

Just like the simply typed λ-calculus, λ_μ satisfies the main meta theoretical properties. We treat these properties now.

Lemma 4.1. *λ_μ is confluent. That is, if $t_1 \twoheadrightarrow t_2$ and $t_1 \twoheadrightarrow t_3$, then there exists a term t_4 such that $t_2 \twoheadrightarrow t_4$ and $t_3 \twoheadrightarrow t_4$.*

Parigot's original proof sketch [Par92], which is based on the notion of parallel reduction by Tait and Martin-Löf, is wrong (this was first noticed by Fujita in [Fuj97]). As observed in [Fuj97, BHF01], the usual notion of parallel reduction does not extend well to λ_μ: it only allows to prove weak confluence. But since λ_μ is strongly normalizing (Lemma 4.6) we have confluence for well-typed terms by Newman's lemma. However, since confluence is a property that also holds for untyped terms, this result is unsatisfactory. Confluence for untyped λ_μ-terms can be proven by analogy to the proof in [GKM11].

In order to prove that λ_μ satisfies subject reduction we have to prove that each reduction rules preserves typing. Because some of the reduction rules involve structural substitution it is convenient to prove an auxiliary result that structural substitution preserves typing first. To express this property we introduce the notion of a *contextual typing judgment*, notation $\Gamma; \Delta \vdash E : \rho \Leftarrow \sigma$, which expresses that $\Gamma; \Delta \vdash t : \sigma$ implies $\Gamma; \Delta \vdash E[t] : \rho$.

Definition 4.2. *The derivation rules for the* contextual typing judgment $\Gamma; \Delta \vdash E : \rho \Leftarrow \sigma$ *are as shown in Figure 2.*

$$\Gamma; \Delta \vdash \Box : \rho \Leftarrow \rho$$

(a) hole

$$\frac{\Gamma; \Delta \vdash E : \sigma \to \tau \Leftarrow \rho \qquad \Gamma; \Delta \vdash t : \sigma}{\Gamma; \Delta \vdash Et : \tau \Leftarrow \rho}$$

(b) app

Figure 2: The rules for contextual typing judgments in $\lambda_\mu^{\mathbf{T}}$.

Fact 4.3. *Contextual typing judgments do indeed enjoy the intended behavior. That is, if $\Gamma; \Delta \vdash E : \rho \Leftarrow \sigma$ and $\Gamma; \Delta \vdash t : \sigma$, then $\Gamma; \Delta \vdash E[t] : \rho$.*

Fact 4.4. *Typing is preserved under (structural) substitution.*

1. *If $\Gamma, x : \rho; \Delta \vdash t : \tau$ and $\Gamma; \Delta \vdash r : \rho$, then $\Gamma; \Delta \vdash t[x := r] : \tau$.*
2. *If $\Gamma; \Delta, \alpha : \rho \vdash t : \tau$ and $\Gamma; \Delta \vdash E : \sigma \Leftarrow \rho$, then $\Gamma; \Delta, \beta : \sigma \vdash t[\alpha := \beta E] : \tau$.*

Proof. The first property is proven by a standard induction on the derivation of $\Gamma, x : \rho; \Delta \vdash t : \tau$. The second property is proven by induction on the derivation of $\Gamma; \Delta, \alpha : \rho \vdash t : \tau$. Most cases are straightforward, so we only consider the passivate case. Let $\Gamma; \Delta, \alpha : \rho \vdash [\alpha]t : \perp\!\!\!\perp$ with $\Gamma; \Delta, \alpha : \rho \vdash t : \rho$. By the induction hypothesis we have $\Gamma; \Delta, \beta : \sigma \vdash t[\alpha := \beta E] : \rho$, which leaves us to prove that $\Gamma; \Delta, \beta : \sigma \vdash ([\alpha]t)[\alpha := \beta E] : \perp\!\!\!\perp$. Since $([\alpha]t)[\alpha := \beta E] \equiv [\alpha]E[t[\alpha := \beta E]]$, the result follows from Fact 4.3 and the induction hypothesis. $\Box$

Lemma 4.5. *λ_μ satisfies subject reduction. That is, if $\Gamma; \Delta \vdash t : \rho$ and $t \to t'$, then $\Gamma; \Delta \vdash t' : \rho$.*

Proof. We have to prove that all reduction rules and the compatible closure preserve typing. We treat some interesting cases.

1. The $\rightarrow_{\mu R}$-rule:

$$\dfrac{\dfrac{\Gamma;\Delta,\alpha:\rho\to\tau\vdash c:\perp\!\!\!\perp}{\Gamma;\Delta\vdash\mu\alpha.c:\rho\to\tau}\quad\Gamma;\Delta\vdash s:\rho}{\Gamma;\Delta\vdash(\mu\alpha.c)s:\tau}\ \rightarrow_{\mu R}\ \dfrac{\Gamma;\Delta,\beta:\tau\vdash c[\alpha:=\beta\ (\Box s)]:\perp\!\!\!\perp}{\Gamma;\Delta\vdash\mu\beta.c[\alpha:=\beta\ (\Box s)]:\tau}$$

   Here we have $\Gamma;\Delta,\beta:\tau\vdash c[\alpha:=\alpha\ (\Box s)]:\perp\!\!\!\perp$ by Fact 4.4 and the fact that $\Gamma;\Delta\vdash\Box s:\rho\to\tau\Leftarrow\tau$

2. The $\rightarrow_{\mu\eta}$-rule:

$$\dfrac{\dfrac{\Gamma;\Delta,\alpha:\rho\vdash t:\rho}{\Gamma;\Delta\vdash[\alpha]t:\perp\!\!\!\perp}}{\Gamma;\Delta\vdash\mu\alpha.[\alpha]t:\rho}\ \rightarrow_{\mu\eta}\ \Gamma;\Delta\vdash t:\rho$$

   Here we have $\Gamma;\Delta\vdash t:\rho$ by strengthening because $\alpha\notin\mathrm{FV}(t)$.

3. The $\rightarrow_{\mu i}$-rule:

$$\dfrac{\dfrac{\Gamma;\Delta,\alpha:\rho,\beta:\rho\vdash c:\perp\!\!\!\perp}{\Gamma;\Delta,\alpha:\rho\vdash\mu\beta.c:\rho}}{\Gamma;\Delta,\alpha:\rho\vdash[\alpha]\mu\beta.c:\perp\!\!\!\perp}\ \rightarrow_{\mu i}\quad\Gamma;\Delta,\alpha:\rho\vdash c[\beta:=\alpha\ \Box]:\perp\!\!\!\perp$$

   Here we have $\Gamma;\Delta,\alpha:\rho,\alpha:\rho\vdash c[\beta:=\alpha\ \Box]:\perp\!\!\!\perp$ by Fact 4.4 and the fact that $\Gamma;\Delta,\alpha:\rho\vdash\Box:\rho\Leftarrow\rho$ □

Lemma 4.6. *λ_μ is strongly normalizing. That is, for all terms t such that $\Delta;\Gamma\vdash t:\rho$, all reduction sequences starting from t are finite.*

Proof. This is proven in [Par97]. □

References

[BHF01] Kensuke Baba, Sachio Hirokawa, and Ken-etsu Fujita. Parallel Reduction in Type Free λ_μ-calculus. *Electronic Notes in Theoretical Computer Science*, 42, 2001.

[FF86] Matthias Felleisen and Daniel P. Friedman. Control operators, the SECD-machine, and the λ-calculus. In Martin Wirsing, editor, *3rd Working Conference on the Formal Description of Programming Concepts*, pages 193–219. North-Holland Publishing, 1986.

[Fuj97] Ken-etsu Fujita. Calculus of Classical Proofs I. In R. K. Shyamasundar and Kazunori Ueda, editors, *ASIAN*, volume 1345 of *LNCS*, pages 321–335. Springer, 1997.

[GKM11] Herman Geuvers, Robbert Krebbers, and James McKinna. The $\lambda_\mu^{\mathbf{T}}$-calculus. Submitted, 2011. Obtained from `http://robbertkrebbers.nl/research/articles/lambda_muT.pdf` on Nov 1, 2011.

[Gri90] Timothy G. Griffin. A Formulae-as-Types Notion of Control. In *POPL*, pages 47–58. ACM, 1990.

[Par92] Michel Parigot. λ_μ-calculus: An Algorithmic Interpretation of Classical Natural Deduction. In Andrei Voronkov, editor, *LPAR*, volume 624 of *LNCS*, pages 190–201. Springer, 1992.

[Par97] Michel Parigot. Proofs of Strong Normalisation for Second Order Classical Natural Deduction. *Journal of Symbolic Logic*, 62(4):1461–1479, 1997.

[RS94] Jakob Rehof and Morten Heine Sørensen. The λ_Δ-calculus. In Masami Hagiya and John C. Mitchell, editors, *TACS*, volume 789 of *LNCS*, pages 516–542. Springer, 1994.