# Towards an atomic lambda-mu-calculus

Fanny He
F.He@bath.ac.uk
University of Bath, Claverton Down
Bath BA2 7AY, United Kingdom

**Abstract.** We construct a $\lambda\mu$-calculus with explicit sharing and atomic reduction, the *atomic* $\lambda\mu$*-calculus*, which is a refinement of the $\lambda\mu$-calculus implementing smaller steps of reduction on individual constructors. We want to study explicit sharing in a calculus with control operators using the concept of atomicity, which comes from deep inference.

## 1 Introduction and background

The methodology of deep inference [6] allows to apply inference rules inside formulas, at arbitrary depth. This gives proof systems with interesting features such as *atomicity*, where rules can be replaced by their restriction on atomic formulas, while preserving essential properties such as cut-elimination [11, 3]. With a computational approach in mind, a $\lambda$-calculus with atomicity was developed to be in Curry-Howard correspondence with an intuitionistic logical system with deep inference in [7]. It showed how atomicity is related to optimal reduction graphs [8], and by giving a fine control over duplications during a $\beta$-reduction, allows to implement a fully lazy sharing reduction strategy [1]. To study the computational content of atomicity in classical logic amounts to adding control operators, which correspond to classical constructs [5] as in the $\lambda\mu$-calculus [9], an extension of the $\lambda$-calculus corresponding to classical natural deduction. In this abstract we introduce the atomic $\lambda\mu$-calculus, encompassing both classical constructs and atomicity.

### 1.1 Deep inference and atomicity

An atomic calculus is a computational interpretation of the transformation $\rightsquigarrow$, which encapsulates the idea behind *open deduction* and atomicity by using the contraction rule $\triangle$, and a linearized version of distributivity, the *medial* rule $m$:

$$\triangle\frac{A}{A\wedge A} \qquad m\frac{(A_1\wedge A_2)\vee(B_1\wedge B_2)}{(A_1\vee B_1)\wedge(A_2\vee B_2)} \qquad \triangle\frac{A\vee B}{(A\vee B)\wedge(A\vee B)} \rightsquigarrow \begin{array}{c}\triangle\dfrac{A}{A\wedge A}\ \vee\ \triangle\dfrac{B}{B\wedge B}\\ \hdashline m\dfrac{(A\wedge A)\vee(B\wedge B)}{(A\vee B)\wedge(A\vee B)}\end{array}$$

In this transformation, we have two proof derivations from the assumption $A\vee B$ to the conclusion $(A\vee B)\wedge(A\vee B)$. The open deduction proof on the right shows how inference rules are restricted to smaller formulas, and exhibits the bidimensional aspect of composition, since derivations can be composed horizontally with connectives (the top-right $\vee$ in our example), and vertically (the dotted lines).

By repeatedly making transformations on smaller subformulas, we ultimately get a derivation where the rules are only applied to atomic formulas. In the example above, this atomicity property allows us to eventually replace the contraction rule $\triangle$ by its atomic version, which is not possible in the sequent calculus [2].

### 1.2 The atomic $\lambda$-calculus

A $\lambda$-calculus with explicit sharing, the *atomic $\lambda$-calculus* [7], was developed as a first computational interpretation via Curry-Howard of a deep inference system.

The atomic $\lambda$-calculus provides a refinement of the $\lambda$-calculus and extends it with an explicit *sharing* constructor (denoted by $[\cdots \leftarrow \bullet]$) corresponding to the contraction rule (similarly to explicit substitutions), a *distributor* constructor (denoted by $[\cdots \twoheadleftarrow \bullet]$) which is an interpretation of the medial rule, and the use of unique variable names such that the $\beta$-reduction is implemented by a *linear* substitution while duplications are performed atomically with *sharings* and *distributors*. Instead of the usual $\beta$-step $(\lambda x.u)t \longrightarrow_\beta u\{t/x\}$, which substitutes the term $t$ for each of the $p$ occurrences of the variable $x$ in $u$, the result of a reduction step in the atomic case is $u[x_1, \ldots, x_p \leftarrow t]$, where $t$ is bound to the variables $x_1, \ldots, x_p$ representing the distinct occurrences of $x$. The duplication of $t$ is then carried out atomically, one constructor at a time, by separate rules.

For example, from $u[x_1, \ldots, x_p \leftarrow \lambda y.v]$ where $\lambda y.v$ is shared by $x_1, \ldots, x_p$ in the term $u$, we want to eventually substitute $x_1, \ldots, x_p$ by $\lambda y.v$, and therefore need to get $p$ copies of $\lambda y.v$. To do this, the idea is to independently obtain copies of the body $v$ and of the constructor $\lambda y$. First, we *freeze* $\lambda y$ (as indicated by $\twoheadleftarrow$) while replicating $v$ $p$-times in a tuple $\langle v_1, \ldots, v_p\rangle$, then perform the substitution by distributing $\lambda y$ over the copies of $v$ to obtain $p$ copies of $\lambda y.v$.

$$\begin{aligned}(\lambda x.u[x_1, \ldots, x_p \leftarrow x])\ \lambda y.v &\rightsquigarrow_\beta u[x_1, \ldots, x_p \leftarrow \lambda y.v] \\ &\rightsquigarrow_* u[x_1, \ldots, x_p \twoheadleftarrow \lambda y.\langle v_1, \ldots, v_p\rangle[y_1, \ldots, y_p \leftarrow y]] \\ &\rightsquigarrow \ u\{(\lambda y_1.v_1)/x_1\}\ \ldots\ \{(\lambda y_p.v_p)/x_p\}\end{aligned}$$

The reduction above gives a computational interpretation of atomicity, and is the main innovation of this calculus. In the atomic $\lambda$-calculus one can perform duplications of subterms independently of their context, in an approach reminiscent of optimal reduction graphs [8]. This corresponds to using the medial rule to perform contractions on smaller formulas.

## 2 The atomic $\lambda\mu$-calculus

We introduce the atomic $\lambda\mu$-calculus to extend the properties of the atomic $\lambda$-calculus to a classical setting. However we must overcome two difficulties to design this calculus. First we need to adapt the structural rule from the $\lambda\mu$-calculus to our setting, then we want to find an similar way to duplicate $\mu$-abstractions, copying independently the $\mu$-constructor and the body of the abstraction.

### 2.1 The $\lambda\mu$-calculus

The $\lambda\mu$-calculus extends the $\lambda$-calculus by adding a *$\mu$-abstraction* constructor:

$$\lambda\mu: \quad t, u ::= x \mid \lambda x.t \mid (t)u \mid \mu\alpha.(t)\beta$$

In this calculus, terms $t$, $u$ are called *unnamed* terms, and subterms of the form $(t)\beta$ are referred to as *named* terms, denoted by $n$ (i.e. $t$ is *named* by $\beta$). A structural rule to reduce applications of $\mu$-abstractions to terms is added to the calculus. Intuitively this rule allows to apply operations on subterms, by passing the argument $t$ only to the subterms that have been named by the $\mu$-variable $\alpha$:

$$(\mu\alpha.n)t \to_\mu \mu\alpha.n\{(w)\, t\alpha/(w)\alpha\}$$

### 2.2 A $\lambda\mu$-calculus with streams and explicit sharings

The substitution $\{(w)\, t\alpha/(w)\alpha\}$ performed after $\to_\mu$ cannot be directly expressed with an explicit substitution. Considering an explicit substitution construction such as $\langle\ (w)\alpha := (w)\, t\alpha\ \rangle$ would be problematic since the *pattern-matching* is made on all subterms of the form $(w)\alpha$. We face the same difficulties with explicit sharings. Consider the following example $v$ with $\beta$ occurring three times in $n$, and its atomic translation $\mathbb{V}$ where all variables occur linearly:

$$v = (\mu\beta.n)\, t_1\, t_2 \overset{atomic}{\longmapsto} \mathbb{V} = (\mu\beta.\, \mathbb{N}\,[\beta_1, \beta_2, \beta_3 \leftarrow \beta])\, \mathbb{T}_1\, \mathbb{T}_2$$

We now need to translate the reduction steps $v \to_\mu (\mu\beta.n\{(w)\, t_1\, \beta/(w)\beta\})t_2 \to_\mu \mu\beta.n\{(w)\, t_1\, t_2\, \beta/(w)\beta\}$.

The $\to_\mu$-rule modifies the structure of all subterms of the form $(w)\beta$, therefore we cannot directly express this reduction in terms of sharings of the form $[var_1, \ldots, var_k \leftarrow subexpression]$.

To solve this problem, we store the whole list of arguments in a right-associative list or *stream* (the operator is denoted by $\circ$), such that:

$$\mathbb{V} \to_\mu (\mu\beta.\, \mathbb{N}\,[\beta_1, \beta_2, \beta_3 \leftarrow \mathbb{T}_1 \circ \beta])\mathbb{T}_2 \to_\mu \mu\beta.\, \mathbb{N}\,[\beta_1, \beta_2, \beta_3 \leftarrow \mathbb{T}_1 \circ (\mathbb{T}_2 \circ \beta)]$$

We therefore work on a variant of the $\lambda\mu$-calculus, the $\lambda\mu S$-calculus [10, 4], which considers right-associative *stream applications* $\circ$:

$$\textit{Streams } S, T ::= \alpha \mid t \circ S \qquad\qquad \textit{Terms } t, u ::= x \mid \lambda x.t \mid (t)u \mid \mu\alpha.(t)S$$

The rule $\to_\mu$ then becomes $(\mu\beta.(t)S)u \longrightarrow_{\mu_t} \mu\beta.((t)S)\{(u \circ \beta)/\beta\}$. We then extend the $\lambda\mu$-calculus with streams to a calculus with explicit sharings:

$$\begin{aligned}
&\textit{Sharings } [\phi], [\psi] ::= [x_1, \ldots, x_p \leftarrow t] \mid [\gamma_1, \ldots, \gamma_p \leftarrow S] \\
&\textit{Streams } S, T ::= \alpha \mid t \circ S \mid (t \circ S)[\phi] \\
&\textit{Names } n, m ::= (t)S \mid n[\phi] \\
&\textit{Terms } t, u ::= x \mid \lambda x.t \mid (t)u \mid \mu\alpha.n \mid u[\phi]
\end{aligned}$$

### 2.3 The atomic $\lambda\mu$-calculus $\lambda\mu S_a$

We get the atomic $\lambda\mu$-calculus by adding a distributor constructor $[\cdots \Leftarrow \bullet]$ for $\mu$-abstractions. Similarly to $\lambda$-distributors $[\cdots \twoheadleftarrow \bullet]$, the $\mu$-distributor enables to duplicate the $\mu$-constructor independently of the body inside a $\mu$-abstraction.

Because of the distinction between terms and streams, the general form of expressions in $\lambda\mu S_a$ is more complicated and the duplication steps explicitly

separate the sharing of terms and streams. However, the idea behind the duplication remains the same, since we first duplicate the body (corresponding to $(t)S$), then distribute the $\mu$-abstraction over the copies of $(t)S$. Sharings and distributors $[\phi_1], \dots, [\phi_r]$ are reduced until all occurrences $\alpha_1, \dots, \alpha_p$ representing $\alpha$ are gathered in the sharing $[\alpha_1, \dots, \alpha_p \leftarrow \alpha]$, then the remaining $[\phi'_1] \dots [\phi'_{r'}]$ can be pushed outside of the distributor before performing the substitution.

$$
\begin{array}{c}
u[x_1, \dots, x_p \leftarrow \mu\alpha.((t)S\,[\phi_1] \dots [\phi_r])] \\
\downarrow_{*} \\
u[x_1, \dots, x_p \Leftarrow \mu\alpha.\langle (t_1)S_1, \dots, (t_p)S_p\rangle[\alpha_1, \dots, \alpha_p \leftarrow \alpha]][\phi'_1] \dots [\phi'_{r'}] \\
\downarrow \\
(u\{(\mu\alpha_1.(t_1)S_1)/x_1\} \ \dots \ \{(\mu\alpha_p.(t_p)S_p)/x_p\})[\phi'_1] \dots [\phi'_{r'}]
\end{array}
$$

## 3 Results and next steps

Following the idea of the atomic $\lambda$-calculus, we have constructed a classical calculus with explicit sharings that satisfies atomicity. Moreover, a type system for the atomic $\lambda\mu$-calculus can also be given using the sequent calculus, with the main new rule being the introduction of the $\mu$-distributor.

The next step is to prove properties of the small-step reduction. Our calculus is built upon the atomic $\lambda$-calculus, which keeps the fundamental properties of the $\lambda$-calculus. We thus expect to retrieve the main properties of the $\lambda\mu$-calculus such as confluence, then strong normalisation and subject reduction in a typed setting, and to preserve strong normalisation with respect to the $\lambda\mu$-calculus.

**Acknowledgement.** I am deeply grateful for insightful discussions to Willem Heijltjes, Guy McCusker and Valentin Blot, to Andrea A. Tubella and Ben Ralph for their helpful comments, and to the reviewers for their useful and detailed remarks to improve this abstract.

## References

1. T. Balabonski. A unified approach to fully lazy sharing. In *POPL*, pages 469–480, 2012.
2. K. Brünnler. Two restrictions on contraction. *Logic Journal of the IGPL*, 11(5):525–529, 2003.
3. K. Brünnler and A. Tiu. A local system for classical logic. In *LPAR*, 2001.
4. M. Gaboardi and A. Saurin. A foundational calculus for computing with streams. In *ICTCS*, 2010.
5. T.G. Griffin. A formulae-as-types notion of control. In *POPL*, 1990.
6. A. Guglielmi. A system of interaction and structure. *CoRR*, cs.LO/9910023, 1999.
7. T. Gundersen, W. Heijltjes, and M. Parigot. Atomic lambda calculus: A typed lambda-calculus with explicit sharing. In *LICS*, 2013.
8. J. Lamping. An algorithm for optimal lambda calculus reduction. In *POPL*, pages 16–30, 1990.
9. M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. *LPAR*, 624:190–201, 1992.
10. A. Saurin. Typing streams in the $\Lambda\mu$-calculus. *ACM Trans. Comp. Logic*, 11(4):28:1–28:34, 2010.
11. A. Tiu. A local system for intuitionistic logic. In *LPAR*, 2006.