Deep-inference intersection types

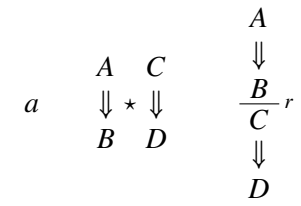
Joseph W.N. Paulus
University of Bath
Claverton Down
Bath Ba2 7AY
jwnp20@bath.ac.uk

Willem Heijltjes
University of Bath
Claverton Down
Bath Ba2 7AY
w.b.heijltjes@bath.ac.uk

Deep inference, as a proof system, has remarkable properties: quasi-polynomial normalization for propositional classical logic; non-elementary proof compression for first-order classical logic; the ability to express logics for which no sequent calculus can exist. It is a natural question if these and other striking features can be put to computational use.

In this extended abstract we will discuss our recent progress in developing a natural treatment of intersection types in deep-inference proof theory, using the variant *open deduction* [3]. Our motivation is two-fold. More broadly, open deduction is a semantically oriented formalism, whose syntax is intimately related to category theory, and we are interested in viewing intersection types from this perspective. More specifically, we are interested in developing intersection types for the *atomic lambda-calculus* [4], a fully lazy lambda-calculus derived as a Curry-Howard interpretation of an open-deduction proof system.

The defining characteristic of open deduction is that proofs are constructed in two dimensions, as illustrated on the right. Left, a type variable a constitutes an *atomic* derivation from premise a to conclusion a. Middle, derivations from A to B and from C to D are composed *horizontally* with a logical connective $\star$, forming a derivation from $A \star C$ to $B \star D$. On the right, the same derivations are composed *vertically* to one from A to D using a (given) inference rule from B to C. Proofs do not branch out, and as a consequence offer interesting opportunities for the sharing of subproofs, or in a computational interpretation, the sharing of subterms.



Our main contribution is as simple as it is elegant: we extend the treatment of connectives in open deduction, as operating at the level of proofs as well as that of formulas, to a (non-idempotent) intersection type constructor.

Intersection types are an important typing system for a calculus, that allows a more comprehensive typing of strongly normalizing terms than a simply typed calculus, as it allows for typing of terms such as $\lambda x.xx$. This is due to allowing variables to have multiple types, hence in this example x can have two different types a and $a \to a$. At the point of abstraction $\lambda x.$ it has both; but in the application xx the type of the function is specialized to $a \to a$, and that of the argument to a.

As our term calculus, we will use a linear lambda-calculus with explicit sharing, named the ***basic calculus*** because it fulfills that role in the development of the atomic lambda-calculus [4]. The calculus is defined by the following grammar, where variables are ***linear***, i.e. they may occur only once, and a bound variable must occur.

$$t, u, v ::= x \mid \lambda x.t \mid tu \mid u[x, y \leftarrow t] \mid u[\leftarrow t]$$

Submitted to:
TYDI 2018

We call these constructors ***variable***, ***abstraction***, ***application***, ***sharing*** and ***weakening*** respectively. We abbreviate a sharing or weakening with $[\phi]$, and a sequence of these by $[\Phi] = [\phi_1]\dots[\phi_n]$. Substitution in the calculus is always linear; we write it as $\{u/x\}$.

Reduction for the basic calculus is similar that of the Linear Substitution Calculus of Accattoli and Kesner [1]. We have (linear) β-reduction, where we incorporate sharings and weakenings into the pattern so that they cannot obstruct a redex, and a duplication and deletion step that evaluate sharings and weakenings, maintaining the linearity of variables. Below, t' and t'' are t with every (free and bound) variable z renamed to a fresh variable z' respectively z'', and the free variables of t are $z_1,\dots,z_n$.

$$\begin{aligned}(\lambda x.t)[\Phi]u &\rightsquigarrow_\beta \lambda x.t\{u/x\}\\ u[x,y\leftarrow t] &\rightsquigarrow_s u\{t'/x\}\{t''/y\}[z'_1,z''_1\leftarrow z_1]\dots[z'_n,z''_n\leftarrow z_n]\\ u[\leftarrow t] &\rightsquigarrow_w u[\leftarrow z_1]\dots[\leftarrow z_n]\end{aligned}$$

Our starting point are non-idempotent intersection types for the lambda calculus, conveniently formulated by Bucciarelli, Kesner and Ventura [2] as ***multiset types***. A multiset type can be of the form $[a,b,a]$, which means that the term has either type a, or b, or a again. Non-idempotent intersections are advantageous over idempotent intersection types as they simplify strong normalization proofs, due to the fact that reductions strictly reduce the size of the type derivation. We define the following types.

$$A,B,C ::= a \mid M\to A \qquad\qquad M,N ::= [A_1,\dots,A_n] \qquad\qquad \Gamma,\Delta ::= M_1\wedge\dots\wedge M_n$$

Here a is a ***type variable***, A is a ***strict*** type, M a ***multiset*** type (a finite multiset of strict types), and Γ a ***context*** type (a finite conjunction of multiset types). We admit the empty context type, a 0-ary conjunction written $\top$, and consider conjunction modulo symmetry, associativity, and unit laws.

Our open-deduction calculus for idempotent intersection types is then built as follows. A typing judgement is of the form $\Gamma\vdash t:A$, and is derived by an open-deduction derivation from Γ to A, as on the right. Here we assume an implicit correspondence between $\Gamma = M_1\wedge\dots\wedge M_n$ and the free variables $x_1,\dots,x_n$ of t, which we may express by $x_i:M_i$ or by $M_i^{x_i}$.

$$\begin{array}{c}\Gamma\\ \Downarrow t\\ A\end{array}$$

Derivations are built following the structure of a term in the basic calculus, within a larger space of constructions for horizontal composition and rules for vertical composition. In isolation, these do not necessarily follow the definition of types and the type pattern of derivations (though they do use the given constructors), but when built from terms, they will.

We employ the following rules, which from left to right are ***axiom***, ***abstraction***, ***application***, ***sharing***, and ***weakening***, and ***medial*** or ***interchange***, where $M+N$ is multiset union.

$$\frac{[A]}{A}\,ax \qquad \frac{\Gamma}{M\to(\Gamma\wedge M)}\,\lambda \qquad \frac{(M\to A)\wedge M}{A}\,@ \qquad \frac{M+N}{M\wedge N}\,\triangle \qquad \frac{M}{\top}\,w \qquad \frac{(M+N)\ \wedge\ (\Gamma+\Delta)}{(M\wedge\Gamma)\ +\ (N\wedge\Delta)}\,x$$

Vertical composition of derivations is as described previously, where the connecting rule between the two derivations can be any of the above. Horizontal composition is with the connectives $\wedge$ and $\to$, and as finite multisets of derivations, as follows. The conclusion of the latter is the multiset $M=[A_1,\dots,A_n]$, and the premise is the multiset over the Δ_i.

$$\begin{array}{ccc}\Gamma & \Delta\\ \Downarrow & \Downarrow\\ A & B\end{array}\wedge \qquad\qquad \begin{array}{c}\Gamma\\ \Downarrow\\ A\end{array}\to M \qquad\qquad \begin{bmatrix}\Delta_1 & \Delta_n\\ \Downarrow & \Downarrow\\ A_1 & A_n\end{bmatrix},\dots,$$

Within this larger space of derivations we can find those that type the terms of the basic calculus. We distinguish the ***strict*** derivations ($\Downarrow$) with a strict type as conclusion, and the ***multiset*** derivations ($\Mapsto$) with multiset type conclusion. Multiset derivations for a term t are generated from the empty and singleton multiset as follows, where the double rule indicates any number of interchange inferences, and $x : M_i$ if and only if $x : N_i$ for each free variable x of t.

$$\dfrac{\dfrac{(M_1 + N_1) \wedge \ldots \wedge (M_n + N_n)}{M_1 \wedge \cdots \wedge M_n \qquad N_1 \wedge \cdots \wedge N_n}}{\begin{array}{ccc} \Downarrow t & + & \Downarrow t \\ M & & N \end{array}}$$

Then strict derivations are constructed as follows.

$$x : \quad \dfrac{[A]^x}{A}\, ax \qquad \lambda x.t : \quad \dfrac{\Gamma}{\dfrac{\Gamma \wedge M^x}{M \to \begin{array}{c}\Downarrow t \\ A\end{array}}\,\lambda} \qquad tu : \quad \dfrac{\begin{array}{ccc}\Gamma & & \Delta \\ \Downarrow t & \wedge & \Downarrow u \\ M \to A & & M\end{array}}{\mathrm{A}}\,@$$

$$u[\leftarrow t] : \quad \left(\begin{array}{c}\Gamma \\ \Delta \wedge \begin{array}{c}\Downarrow t \\ M \\ \hline \top\end{array} w\end{array}\right) \qquad u[x,y \leftarrow t] : \quad \left(\begin{array}{c}\Gamma \\ \Delta \wedge \begin{array}{c}\Downarrow t \\ M + N \\ \hline M^x \wedge N^y\end{array} \triangle\end{array}\right)$$
$$\Downarrow u \qquad\qquad \Downarrow u$$
$$A \qquad\qquad A$$

The proposed calculus enjoys subject reduction. It gives a purely structural treatment of intersection types, where all necessary logical components are immediately visible. We expect that the principle of lifting an intersection type constructor to the level of derivations will apply straightforwardly to idempotent types, and more broadly when the *medial* rule is available. This is the topic of future investigations.

References

[1] Beniamino Accattoli and Delia Kesner. The structural lambda-calculus. In *International Workshop on Computer Science Logic (CSL)*, 2010.

[2] Antonio Bucciarelli, Delia Kesner, and Daniel Ventura. Non-idempotent intersection types for the lambda-calculus. *Logic Journal of the IGPL*, 25(4):431–464, 2017.

[3] Alessio Guglielmi, Tom Gundersen, and Michel Parigot. A proof calculus which reduces syntactic bureaucracy. In *Rewriting Techniques and Applications (RTA)*, pages 135–150, 2010.

[4] Tom Gundersen, Willem Heijltjes, and Michel Parigot. Atomic lambda-calculus: a typed lambda-calculus with explicit sharing. In *28th IEEE Symposium on Logic in Computer Science LICS*, pages 311–320, 2013.