The Next 700 Modal Type Assignment Systems

Andreas Abel

Department of Computer Science and Engineering
Gothenburg University

We exhibit a generic modal type system for simply-typed lambda-calculus that subsumes linear and relevance typing, strictness analysis, variance (positivity) checking, and other modal typing disciplines. By identifying a common structure in these seemingly unrelated non-standard type systems, we hope to gain better understanding and a means to combine several analyses into one. This is work in progress.

Our modal type assignment system is parametrized by a (partially) ordered monoid (P, $\_\cdot\_$, $\mathbb{1}$, $\leq$) with a partial, monotone binary operation $\_+\_$ and a default element $p_0 \in P$. Types T, U include at least a greatest type $\top$ and function types $Q \to T$, and form a partial ordering under subtyping $T \leq T'$ with partial meet $T \wedge T'$. Modal types $Q ::= pT$ support composition pQ and partial meet $Q \wedge Q'$ defined by $p(qT) = (pq)T$ and $pT \wedge qT = (p+q)T$. Subtyping $pT \leq p'T'$ holds if $p \leq p'$ and $T \leq T'$.

For typing contexts Γ, Δ, which are total functions from term variables to modal types, modality composition $p\Gamma$, subsumption $\Gamma \leq \Delta$, and meet $\Gamma \wedge \Delta$ are defined pointwise. Finite contexts $x_1 \!:\! Q_1, \dots, x_n \!:\! Q_n$ are represented as $\Gamma(x_i) = Q_i$ and $\Gamma(y) = p_0\top$ for $y \neq x_i$.

Judgements $\Gamma \vdash t : T$ and $\Gamma \vdash t : Q$ are given by the following (linear) typing rules:

$$\frac{p \leq \mathbb{1}}{x : pT \vdash x : T}\ \textsc{hyp} \qquad \frac{\Gamma, x : Q \vdash t : T}{\Gamma \vdash \lambda x t : Q \to T}\ \textsc{abs} \qquad \frac{\Gamma \vdash t : Q \to T \qquad \Delta \vdash u : Q}{\Gamma \wedge \Delta \vdash t\,u : T}\ \textsc{app}$$

$$\frac{\Gamma \leq \Delta \qquad \Delta \vdash t : T \qquad T \leq U}{\Gamma \vdash t : U}\ \textsc{sub} \qquad \frac{\Gamma \vdash t : T}{p\Gamma \vdash t : pT}\ \textsc{mod}$$

The default modality p_0 controls weakening: We can use the subsumption rule SUB with $(\Gamma, x\!: pT) \leq \Gamma$ which holds if $p \leq p_0$ (as then $pT \leq p_0\top = \Gamma(x)$). Meaningful instances of our modal type assignment system abound, here are a few:

1. **Simple typing:** $P = \{1\}$ with $1 + 1 = 1$ and t well-typed if $\Gamma \vdash t : T \neq \top$.
2. **Quantitative typing:** Take some $P \subseteq \mathcal{P}(\mathbb{N})$ closed under $p \cdot q = \{nm \mid n \in p, m \in q\}$ and define $p \leq q$ as $p \supseteq q$ and $p + q$ as $\bigcap\{r \in P \mid r \supseteq \{n+m \mid n \in p, m \in q\}\}$. If $\mathbb{0} := \{0\} \in P$, it is a zero.

   The rule MOD has an intuitive reading in quantitative typing: If t produces a T from resources Γ, we can produce p times T from the p-fold resources $p\Gamma$. Subsumption SUB may allow us to produce less (or the same) from more (or the same) resources. A modal function type $pU \to T$ requires p-fold U to deliver one T.

   Instances of quantitative typing include:

   (a) **Linear typing:** [4] $P = \{\mathbb{0}, \mathbb{1}\}$ with unit $\mathbb{1} = \{1\}$ and default $p_0 = \mathbb{0}$ forbidding weakening with linear variables $x : \mathbb{1}T$ (as $\mathbb{1} \not\leq p_0$). Contraction is also forbidden as $\mathbb{1} + \mathbb{1}$ is undefined.

   (b) **Affine typing:** $P = \{\mathbb{0}, \mathbb{1}\}$ with unit $\mathbb{1} = \{0, 1\}$, allowing weakening as $\mathbb{1} \leq p_0 = \mathbb{0}$.

   (c) **Relevant typing:** $P = \{\mathbb{0}, \mathbb{1}\}$ with unit $\mathbb{1} = \mathbb{N} \setminus \mathbb{0}$, allowing contraction as $\mathbb{1} + \mathbb{1} = \mathbb{1}$.

(d) **Linear and unrestricted hypotheses:** $P = \{!, \mathbb{1}\}$ with $\mathbb{1} = \{1\}$ and $p_0 = ! = \mathbb{N}$. Allows weakening and contraction for $x : !T$.

(e) **Strictness typing:** [2] $P = \{l, s\}$ with *lazy* $p_0 = l = \mathbb{N}$ and unit *strict* $s = \mathbb{N} \setminus \mathbb{0}$. We cannot weaken with strict variables. As $p + q = s$ iff $p = s$ or $q = s$, one strict occurrence of a variable x suffices to classify a function $\lambda xt : sT \to T'$ as strict, whereas a function is lazy only if all occurrences of parameter x are lazy.

3. **Variance (positivity):** $P = \{\emptyset, +, -, \pm\} = \mathcal{P}\{+1, -1\}$ with unit $+ = \{+1\}$ denoting *positive occurrence*, $- = \{-1\}$ *negative occurrence*, $\pm = \{+1, -1\}$ *mixed occurrence*, and $p_0 = \emptyset$ *no occurrence*. With $p \leq q$ iff $p \supseteq q$ and $pq = \{ij \mid i \in p, j \in q\}$ and $p + q = p \cup q$ we obtain *variance typing* aka *positivity checking* for type-level lambda calculi [1].

We can go further and give up the distinction between types and modal types, leading to the types $T, U ::= \top \mid U \to T \mid pT \mid \ldots$ quotiented by $p(qT) = (pq)T$. This makes modal types first class, and we can simplify the hypothesis rule to

$$\frac{}{x : T \vdash x : T} \ \textsc{hyp}.$$

Thus, we subsume further type systems:

1. **Linear typing with exponential:** As 2d, but now $!T$ is a valid type.

2. **Nakano's modality for recursion [3]:** Basic modalities are *later* $\triangleright$ and *always* $\square$ with $\square \cdot p = \square$, generating the modalities $P = \{\triangleright^n, \triangleright^n\square \mid n \in \mathbb{N}\}$ with unit $\mathbb{1} = \triangleright^0$ and partial order $\triangleright^k\square \leq \triangleright^l\square \leq \triangleright^l \leq \triangleright^m$ for $k \leq l \leq m$. Since $x : U \to T, y : U \vdash x\,y : T$ entails $x : \triangleright(U \to T), y : \triangleright U \vdash x\,y : \triangleright T$ by MOD, idiomatic application $\lambda x\lambda y.xy : \triangleright(U \to T) \to \triangleright U \to \triangleright T$ is definable.

Acknowledgments. Thanks to the anonymous referees, who helped improving the quality of this abstract through their feedback. This work was supported by Vetenskapsrådet through the project *Termination Certificates for Dependently-Typed Programs and Proofs via Refinement Types*.

References

[1] Andreas Abel. Polarized subtyping for sized types. *Math. Struct. in Comput. Sci.*, 18:797–822, 2008. Special issue on subtyping, edited by Healfdene Goguen and Adriana Compagnoni.

[2] Stefan Holdermans and Jurriaan Hage. Making "stricterness" more relevant. *J. Higher-Order and Symb. Comput.*, 23(3):315–335, 2010.

[3] Hiroshi Nakano. A modality for recursion. In *Proc. of the 15th IEEE Symp. on Logic in Computer Science (LICS 2000)*, pages 255–266. IEEE Computer Soc. Press, 2000.

[4] David Walker. Substructural type systems. In Benjamin C. Pierce, editor, *Advanced Topics in Types and Programming Languages*, chapter 1. MIT Press, 2005.