Type inference for monotonicity

Michael Arntzenius

University of Birmingham

Problem: Ensuring functions are monotone

Monotonicity crops up in interesting places:

1. In the query languages Datalog and Datafun [3], monotonicity is needed to ensure recursive queries terminate.
2. In abstract interpretation, static analyses are phrased as monotone maps on lattices.
3. For ensuring eventual consistency in distributed systems [2] and determinism in concurrent systems [4].

In all these contexts, it's useful to be able to guarantee a function is monotone. So: **how can we ensure monotonicity using types?**

Modes of (non-)monotonicity

I consider four **modes**, ways a function may respect the order on its domain:

- id is monotone, or order-preserving. For example, $\lambda x.\, x$.
- op is antitone, or order-inverting. For example, not : bool $\to$ bool.
- $\Box$ is equivariant, preserving only equivalence. Usually, *all* functions are equivariant.
- $\Diamond$ is bivariant, or both mono- and antitone. For example, $\lambda x.\, 42$.

Modes as preorder transformations

Formally, modes alter **preorderings** (reflexive, transitive relations), as shown in Figures 1 and 2. We say $f : A \to B$ has mode T iff f is monotone from $TA \to B$.

Modes are *ordered* by what they do to preorders: $T \leqslant U$ iff $x \leqslant y : TA \implies x \leqslant y : UA$. Modes can also be *composed*: If $f : A \to B$ has mode T and $g : B \to C$ has mode U, then $g \circ f$ has mode UT.

Variables get usage modes, too

Besides types, we also care about the *mode* at which a variable is used. To simplify our examples, we consider only single-variable contexts. The typing judgment is then:

$$x : [T]A \vdash M : B$$

Approach 1: Annotate the arrows

Annotating function types with their mode is the obvious approach, used in Datafun [3] and variance typing [1]:

$$\begin{aligned} \text{setMap} &: (A \xrightarrow{\Box} B) \xrightarrow{\Box} \text{Set}\,A \xrightarrow{\text{id}} \text{Set}\,B \\ \text{setMap}\ f\ xs &= \text{do}\ x \leftarrow xs \\ &\qquad \text{return}\ (f\ x) \end{aligned}$$

But it cannot capture more complex input-output ordering relationships:

$$\begin{aligned} \text{subtractEach} &: \text{List}\,(\mathbb{N} \times \mathbb{N}) \xrightarrow{???} \text{List}\,\mathbb{N} \\ \text{subtractEach}\ xs &= \text{map}\ (\lambda(x, y).\, x - y)\ xs \end{aligned}$$

A mode on the arrow cannot indicate this function is **monotone** in the first half of each $(\mathbb{N} \times \mathbb{N})$ pair, but **antitone** in the second.

Approach 2: Modal types

Let $(\text{op}\,A)$ be A with its ordering inverted. **Antitone** maps $(A \xrightarrow{\text{op}} B)$ are just **monotone** maps $(\text{op}\,A \to B)$. Instead of annotating arrows, we can make **all** functions monotone, and apply modes directly to types! Now subtractEach has a precise type:

$$\text{subtractEach} : \text{List}\,(\mathbb{N} \times \text{op}\,\mathbb{N}) \to \text{List}\,\mathbb{N}$$

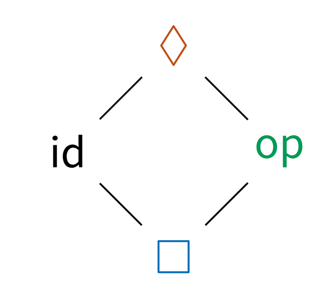
A key feature of modal type systems [5] is that the intro and elim rules for modal types **manipulate the modes in the typing context**:

$$\frac{x : [T]A \vdash M : B}{x : [\Box T] \vdash \text{box}\,M : \Box B} \qquad \frac{x : [T]A \vdash M : \Box B \qquad y : [\Box]B \vdash N : C}{x : [T]A \vdash \text{let box}\,y = M\ \text{in}\ N : C}$$

However, needing to **explicitly introduce and eliminate** modal types clutters up functions like setMap:

$$\begin{aligned} &\text{setMap} : \Box(\Box A \to B) \to \text{Set}\,A \to \text{Set}\,B \\ &\text{setMap}\ f\ xs = \\ &\quad \text{let box}\ g = f\ \text{in} \\ &\quad \text{do}\ x \leftarrow xs \\ &\qquad \text{let box}\ y = x \\ &\qquad \text{return}\ (\text{box}\ (g\ (\text{box}\ y))) \end{aligned}$$

$$\begin{aligned} a \leqslant b : \text{id}\,A &\iff a \leqslant b : A \\ a \leqslant b : \text{op}\,A &\iff a \geqslant b : A \\ a \leqslant b : \Box A &\iff a \leqslant b \wedge a \geqslant b : A \\ a \leqslant b : \Diamond A &\impliedby a \leqslant b \vee b \leqslant a : A \end{aligned}$$



UT	T: id	op	$\Box$	$\Diamond$
U: id	id	op	$\Box$	$\Diamond$
op	op	id	$\Box$	$\Diamond$
$\Box$	$\Box$	$\Box$	$\Box$	$\Diamond$
$\Diamond$	$\Diamond$	$\Diamond$	$\Box$	$\Diamond$

Figure 1: Modes, the mode lattice, and mode composition

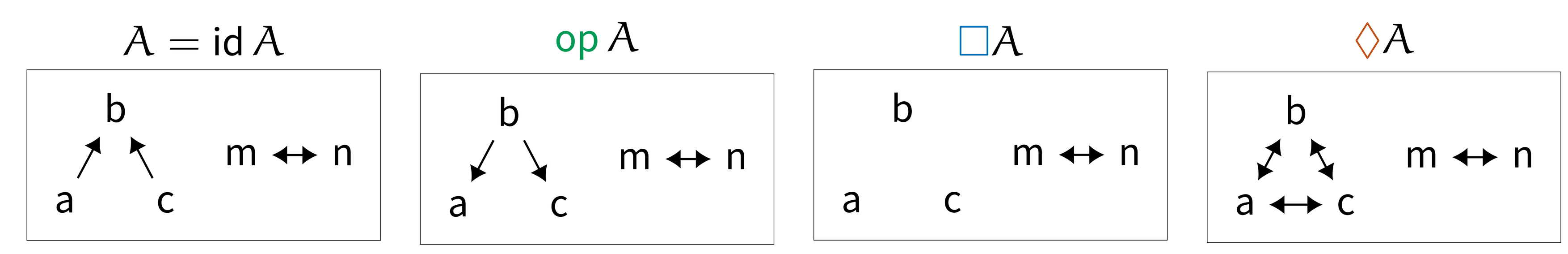


Figure 2: Applying various modes to a preorder

Our approach: Modal subtyping!

Goal: Handle functions which are monotone in only part of their input *without* clunky term annotations:

$$\begin{aligned} \text{subtractEach} &: \text{List}\,(\mathbb{N} \times \text{op}\,\mathbb{N}) \to \text{List}\,\mathbb{N} \\ \text{subtractEach}\ xs &= \text{map}\ (\lambda(x, y).\, x - y)\ xs \end{aligned} \qquad \begin{aligned} \text{setMap} &: \Box(\Box A \to B) \to \text{Set}\,A \to \text{Set}\,B \\ \text{setMap}\ f\ xs &= \text{do}\ x \leftarrow xs; \text{return}\ (f\ x) \end{aligned}$$

Method: Construct and eliminate modal types *implicitly* via **subtyping**. Since types are preorders, subtyping means *subpreordering*:

$$A <: B \text{ iff } A \subseteq B \text{ and } x \leqslant y : A \implies x \leqslant y : B$$

For example, $\Box A <: A$, because $x \leqslant y \wedge y \leqslant x : A \implies x \leqslant y : A$. This lets subtyping *eliminate* $\Box A$. But, how can it *introduce* $\Box$? To do that, just like our intro and elim rules, we must let subtyping **alter the modes in the typing context**.

Modal subtyping alters the context

We *generalize* our subtyping judgment to $[T]A <: B$, giving the **greatest** tone T such that $TA <: B$.

$$\textsc{subsumption}\ \frac{x : [T]A \vdash M : B \qquad [U]B <: C}{x : [UT]A \vdash M : C} \qquad \Box\textsc{-injection}\ \frac{[T]A <: B}{[\Box T]A <: \Box B} \qquad \Box\textsc{-extraction}\ \frac{[T]A <: B}{[T\Diamond]\Box A <: B}$$

Example typing and subtyping rules

I now drop the pretense that contexts have only one variable.

$$T(x_i : [U_i]A_i)_i = (x_i : [TU_i]A_i)_i \qquad \frac{x : [T]A \in \Gamma \quad T \leqslant \text{id}}{\Gamma \vdash x : A} \qquad \frac{\Gamma, y : [U]B \vdash M : C \quad \text{id} \leqslant U}{\Gamma \vdash \lambda y.\, M : B \to C}$$

$$\frac{\Gamma_1 \vdash M : A \quad [T]A <: B \to C \quad \Gamma_2 \vdash N : B}{T\Gamma_1 \wedge \Gamma_2 \vdash M\,N : C} \qquad \frac{\Gamma_1 \vdash M : A \quad \Gamma_2, x : [T]A \vdash N : B}{T\Gamma_1 \wedge \Gamma_2 \vdash \text{let}\ x = M\ \text{in}\ N : B}$$

$$[\text{id}]A <: A \qquad \frac{[T]A <: B}{[UT]A <: UB} \qquad \frac{[T]A <: B \quad U \dashv V}{[TU]VA <: B}$$

$$\frac{U \in \{\text{id}, \text{op}, \Diamond\} \quad U \leqslant T \quad [T]A_2 <: A_1 \quad [U]B_1 <: B_2}{[U](A_1 \to B_1) <: A_2 \to B_2} \qquad \frac{[\Diamond]A_2 <: A_1 \quad [\Box]B_1 <: B_2}{[\Box](A_1 \to B_1) <: A_2 \to B_2}$$

References

[1] Andreas Abel.
Polarized subtyping for sized types.
In Dima Grigoriev, John Harrison, and Edward A. Hirsch, editors, *Computer Science - Theory and Applications, First International Computer Science Symposium in Russia, CSR 2006, St. Petersburg, Russia, June 8-12, 2006, Proceedings*, volume 3967 of *Lecture Notes in Computer Science*, pages 381--392. Springer, 2006.

[2] Peter Alvaro, Neil Conway, Joseph M. Hellerstein, and William R. Marczak.
Consistency analysis in Bloom: a CALM and collected approach.
In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings*, pages 249--260, 2011.

[3] Michael Arntzenius and Neelakantan R. Krishnaswami.
Datafun: A functional Datalog.
In *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming*, ICFP 2016, pages 214--227, New York, NY, USA, 2016. ACM.

[4] Lindsey Kuper and Ryan R. Newton.
LVars: lattice-based data structures for deterministic parallelism.
In Clemens Grelck, Fritz Henglein, Umut A. Acar, and Jost Berthold, editors, *Proceedings of the 2nd ACM SIGPLAN workshop on Functional high-performance computing, Boston, MA, USA, FHPC@ICFP 2013, September 25-27, 2013*, pages 71--84. ACM, 2013.

[5] Frank Pfenning and Rowan Davies.
A judgmental reconstruction of modal logic.
Mathematical Structures in Computer Science, 11(4):511--540, 2001.