SEMINAR:
A Unified View of Modalities in Types Systems

Elvis Rossi

Department of Computer Science

University of Pisa

Master Degree in Computer Science

February 17, 2025

Table of Contents I

Outline

In Philosophy

A modality in philosophy and formally in formal logic/type theory expresses a certain mode (or "moment" as in Hegel) of being. According to Kant the four "categories" are:

- Quantity
- Quality
- Relation
- Modality

and the modalities contain the three pairs of opposites:

- possibility — impossibility
- being — nothing
- necessity — Zufälligkeit

In formal logic and type theory

In formal logic and type theory modalities are formalized by modal operators or closure operators #, that send propositions/types X to new propositions/types $\#X$, satisfying some properties.

Adding such modalities to propositional logic or similar produces what is called modal logic. Here the most famous modal operators are those meant to formalize necessity (denoted $\Box$) and possibility (denoted $\Diamond$), which together form S4 modal logic. Similarly, adding modalities more generally to type theory and homotopy type theory yields modal type theory and modal homotopy type theory

Outline

Ringoid of Modalities

The modality structure is a ring-like structure, which parameterises the calculus Λ^p.
It is a 6-tuple consisting of a set M, addition $(+)$, multiplication $(\cdot)$, meet $(\wedge)$, element 0 and element 1.

- (M, +, 0) forms a commutative monoid (associative and commutative), with 0 as identity

- (M, $+$, 0) forms a commutative monoid (associative and commutative), with 0 as identity
- (M, $\cdot$, 1) forms a monoid (associative), with 1 as identity

- $(M, +, 0)$ forms a commutative monoid (associative and commutative), with 0 as identity
- $(M, \cdot, 1)$ forms a monoid (associative), with 1 as identity
- 0 is an absorbing element for multiplication: $p \cdot 0 = 0 \cdot p = 0$

- (M, $+$, 0) forms a commutative monoid (associative and commutative), with 0 as identity
- (M, $\cdot$, 1) forms a monoid (associative), with 1 as identity
- 0 is an absorbing element for multiplication: $p \cdot 0 = 0 \cdot p = 0$
- (M, $\wedge$) forms a semilattice: meet is associative, commutative and idempotent.

- (M, $+$, 0) forms a commutative monoid (associative and commutative), with 0 as identity
- (M, $\cdot$, 1) forms a monoid (associative), with 1 as identity
- 0 is an absorbing element for multiplication: $p \cdot 0 = 0 \cdot p = 0$
- (M, $\wedge$) forms a semilattice: meet is associative, commutative and idempotent.
- multiplication distributes over addition: $p(q + r) = pq + pr$ and $(p + q)r = pr + qr$
- multiplication distributes over meet: $p(q \wedge r) = pq \wedge pr$ and $(p \wedge q)r = pr \wedge qr$
- addition distributes over meet: $(p \wedge q) + r = (p + r) \wedge (q + r)$

Definition (Order)

$$(p \leq q) \triangleq (p = p \wedge q)$$

Definition (Modality context)

A modality context (or usage map) is defined as a map from variable names to modality expressions.

- Every variable in the judgement $\gamma\Gamma \vdash t : A$ is qualified by a modality.
- t has type A with unit qualification, in context $\gamma\Gamma$
- notation: $\gamma\Gamma, x :^{p} A \triangleq (\gamma, x \mapsto p)(\Gamma, x : A)$

Some properties:

$$(\gamma + \delta)(x) = \gamma(x) + \delta(x),$$
$$(\gamma \wedge \delta)(x) = \gamma(x) \wedge \delta(x),$$
$$(q \cdot \gamma)(x) = q \cdot \gamma(x)$$

Predicative Polymorphic Lambda Calculus with Modalities

Λ^p is a functional programming language with:

- predicative polymorphism $\forall\alpha.B$,
- modal function types ${}^pA \to B$,
- modal boxing $p\langle A\rangle$,
- and modality polymorphism $\forall m.B$.

Types

Definition

Types $A, B, C \in \mathtt{Ty}$ are given by the following grammar:

$$\begin{array}{rcl} \text{A,B,C} & ::= & K \mid 1 \mid \alpha \\ & \mid & \forall \alpha.A \mid \forall m.A \\ & \mid & {}^{p}A \to B \mid A + B \\ & \mid & A \times B \mid p\langle A \rangle \end{array}$$

Let Ty_0 denote the set of monomorphic types (monotypes).
If α is restricted to only monotypes, Λ^p becomes predicative.
Let Ty_0^0 denote the set of closed monotypes.

- If $\gamma\Gamma \vdash A$ then $p\gamma\Gamma$ is needed to produce $p\langle A \rangle$.
- If $\gamma\Gamma \vdash A$ and $\delta\Gamma \vdash B$, then $(\gamma + \delta)\Gamma$ is needed to produce both A and B.

$$^{p}A \to B$$

$$^{p}A \to B$$

$$^{1}A \to B \quad = \quad A \to B \quad = \quad A \multimap B$$

$$^{p}A \to B$$

$$^{1}A \to B \quad = \quad A \to B \quad = \quad A \multimap B$$

$$^{p}A \to B \quad = \quad p\langle A \rangle \to B$$

In a language with generalised algebraic data types, $p\langle A\rangle$ would be defined instead as a data type with constructor of type

$$^{p}A \to p\langle A\rangle$$

Typing rules for Λ^p

System F (*from lambda calculus*)	Λ^p
$\text{VAR}\ \dfrac{}{\Gamma, x : A \vdash x : A}$	$\text{VAR}\ \dfrac{}{0\Gamma, x : {}^{1}A \vdash x : A}$
$\text{ABS}\ \dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}$	$\text{ABS}\ \dfrac{\gamma\Gamma, x : {}^{q}A \vdash t : B}{\gamma\Gamma \vdash \lambda\, {}^{q}x.t : {}^{q}A \to B}$
$\text{APP}\ \dfrac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}$	$\text{APP}\ \dfrac{\gamma\Gamma \vdash t : {}^{q}A \to B \quad \delta\Gamma \vdash u : A}{(\gamma + q\delta)\Gamma \vdash t\ {}^{q}u : B}$

System F

$$\text{T-ABS}\ \frac{(\Gamma, \alpha) \vdash t : B}{\Gamma \vdash \Lambda\alpha.t : \forall\alpha.B}$$

$$\text{T-APP}\ \frac{\Gamma \vdash t : \forall\alpha.B \quad \Gamma \vdash A}{\Gamma \vdash t \cdot A : B[A/\alpha]}$$

Λ^p

$$\text{T-ABS}\ \frac{\gamma(\Gamma, \alpha) \vdash t : B}{\gamma\Gamma \vdash \Lambda\alpha.t : \forall\alpha.B}$$

$$\text{T-APP}\ \frac{\gamma\Gamma \vdash t : \forall\alpha.B \quad \Gamma \vdash A}{\gamma\Gamma \vdash t \cdot A : B[A/\alpha]}$$

$$\Lambda^{p}$$

$$\mathsf{p}\langle\cdot\rangle\text{-INTRO}\,\frac{\gamma\Gamma \vdash t : A}{p\gamma\Gamma \vdash [^{p}t] : p\langle A\rangle}$$

$$\mathsf{p}\langle\cdot\rangle\text{-ELIM}\,\frac{\gamma\Gamma \vdash u : p\langle A\rangle \quad \delta\Gamma, x : {}^{qp}A \vdash t : C}{(q\gamma + \delta)\Gamma \vdash \mathtt{let}[^{p}x] = {}^{q}u \ \mathtt{in}\ t : C}$$

$$\mathsf{WK}\,\frac{\delta\Gamma \vdash t : A \quad \gamma \leq \delta}{\gamma\Gamma \vdash t : A}$$

Theorem (Convertibility)

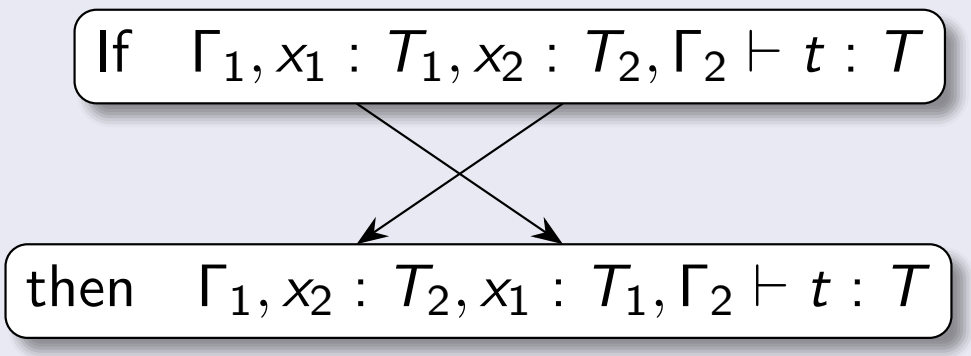
If $p \leq q$, then there is a term of type ${}^{p}A \to q\langle A \rangle$ for any A.

Outline

Consider the three basic structural properties satisfied by simply-typed lambda calculus:

Definition (Exchange)

Exchange indicates that the order in which we write down variables in the context is irrelevant:



Definition (Weakening)

Weakening indicates that adding extra unneeded assumptions to the context does not prevent a term from type checking:

$$\text{If} \quad \Gamma_1, \Gamma_2 \vdash t : T$$

$$\text{then} \quad \Gamma_1, x_1 : T_1, \Gamma_2 \vdash t : T$$

Definition (Contraction)

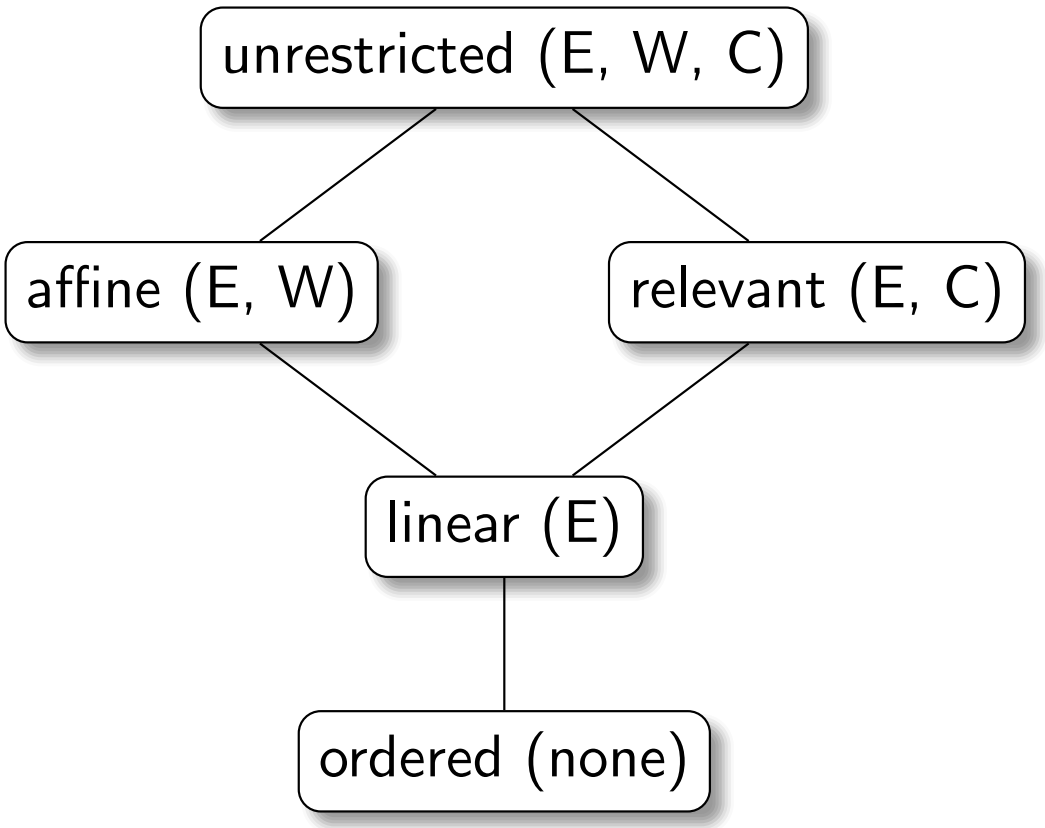
Contraction states that if we can type check a term using two identical assumptions ($x_2 : T_1$ and $x_3 : T_1$) then we can check the same term using a single assumption:

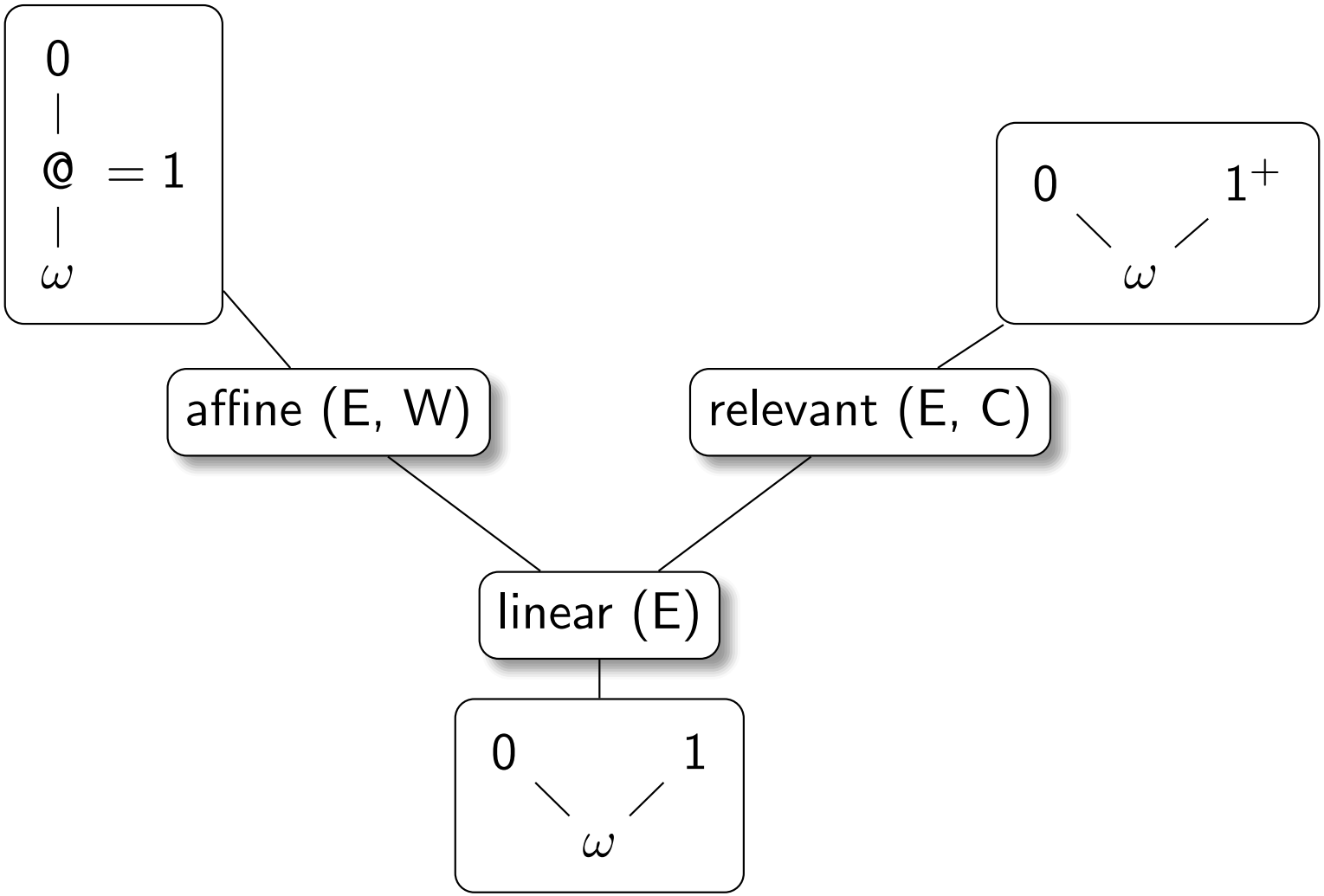
$$\text{If} \quad \Gamma_1, x_2 : T_1, x_3 : T_1, \Gamma_2 \vdash t : T_2$$

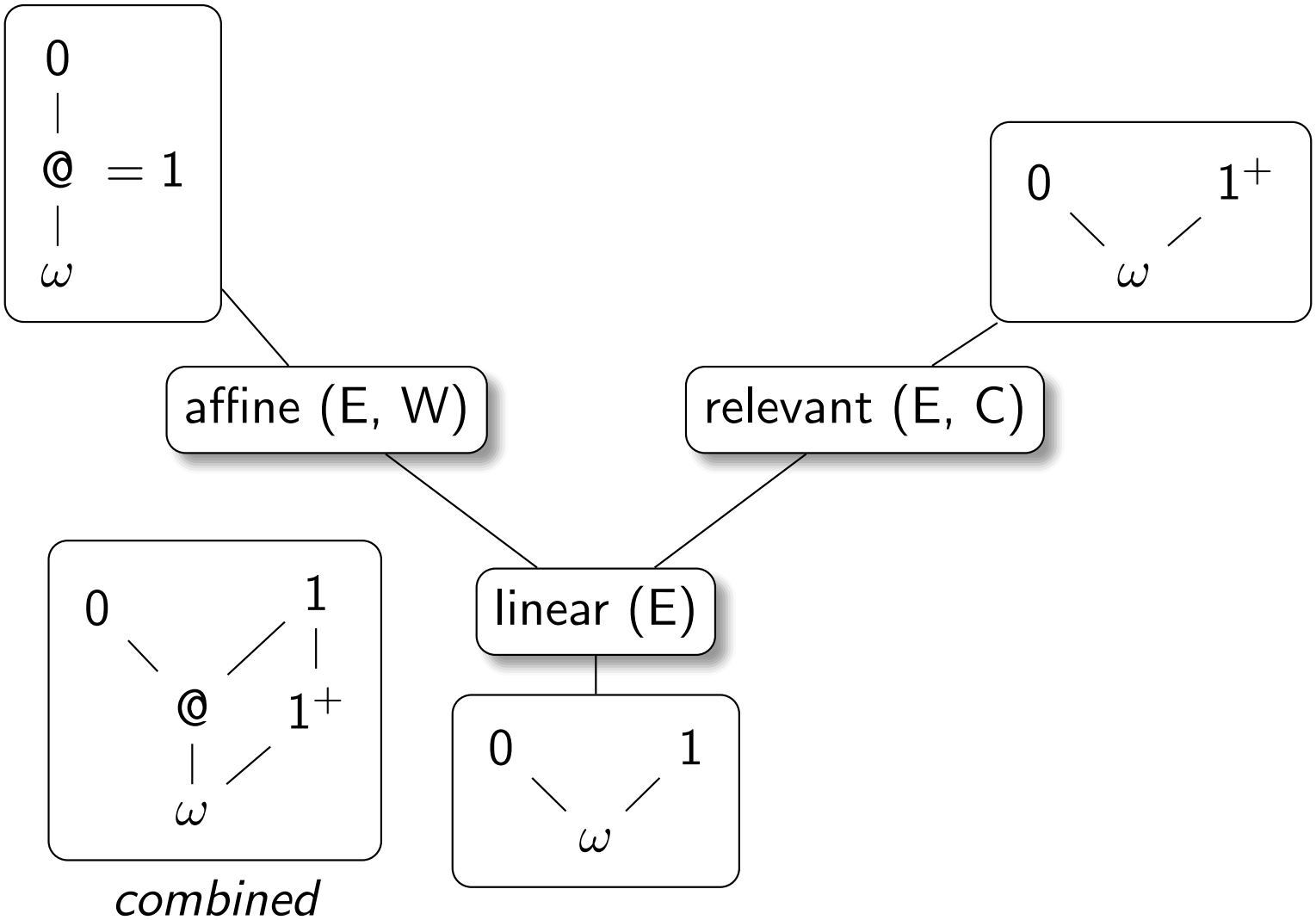
$$\text{then} \quad \Gamma_1, x_1 : T_1, \Gamma_2 \vdash [x_2 \mapsto x_1][x_3 \mapsto x_1]t : T_2$$

We thus have different types systems if we allow different rules to be applied:

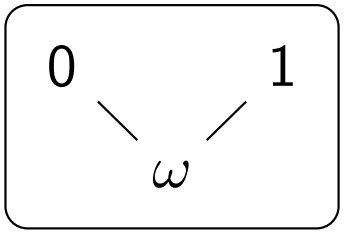
- *Linear* type systems ensure that every variable is used exactly once (E).
- *Affine* type systems ensure that every variable is used at most once (E, W).
- *Relevant* type systems ensure that every variable is used at least once (E, C).
- *Ordered* type systems ensure that every variable is used exactly once and in the order which it is introduced (none).


unrestricted (E, W, C)
affine (E, W)
relevant (E, C)
linear (E)
ordered (none)


0
@
ω
= 1
0
1+
ω
affine (E, W)
relevant (E, C)
linear (E)
0
1
ω


0
@ = 1
ω
0 1+
ω
affine (E, W)
relevant (E, C)
0 1
@ 1+
ω
linear (E)
0 1
ω
combined

Linear Types



Indeed the unit modality precisely corresponds to linear usages.
A 0-qualified function is necessarily constant (*irrelevance*).
ω represents non-linear usage, equivalent to the more traditional exclamation mark (!).

Linear Type System

$$
\begin{array}{rcl}
\mathcal{A} & ::= & p \mid p^{\perp} \\
& \mid & A \otimes A \mid A \oplus A \\
& \mid & A \& A \mid A \parr A \\
& \mid & 1 \mid 0 \mid \top \mid \bot \\
& \mid & !A \mid ?A
\end{array}
$$

The intuitionistic implication $B \implies C$ can be defined as $!B \multimap C$
Where

$$B \multimap A \triangleq B^{\perp} \parr C$$

Finally the rules for $\cdot$ and $+$:

$(+)$	0	ω	$@$	1	1^+
0	0	ω	$@$	1	1^+
ω	ω	ω	ω	1^+	1^+
$@$	$@$	ω	ω	1^+	1^+
1	1	1^+	1^+	1^+	1^+
1^+	1^+	1^+	1^+	1^+	1^+

$(\cdot)$	0	ω	$@$	1	1^+
0	0	0	0	0	0
ω	0	ω	ω	ω	ω
$@$	0	ω	$@$	$@$	ω
1	0	ω	$@$	1	1^+
1^+	0	ω	ω	1^+	1^+

Quantitative Typing

A generalisation of all the above systems in what can be called quantitative typing, with one modality for each set of accepted usage.
Where the set of all modalities Mod $= 2^{\mathbb{N}}$, with $0 = \{0\}, 1 = \{1\}$ and:

$$p + q = \{x + y | x \in p, y \in q\}$$
$$p \cdot q = \{x \cdot y | x \in p, y \in q\}$$
$$p \wedge q = p \cup q$$

Every acceptable set of usage is tracked.

Sensitivity Analysis for Differential Privacy

In differential privacy one is interested in publishing statistically anonymised data without revealing individual secrets.
The role of the type system is to ensure that if a certain amount of noise is introduced in the inputs of a program, then at least the same amount is present in the outputs.

In the literature every type A is equipped with a metric $d_A : A \times A \to \mathbb{R}^{\infty}_{\geq 0}$.
$f : A \to B$ is c-sensitive if it does not increase distances by a factor greater than c:

$$d_B(f(x), f(y)) \leq_{\mathbb{R}} c \cdot d_A(x, y)$$

To translate the model into the proposed framework, let the modality carrier set be $\mathbb{R}^{\infty}_{\geq 0}$; let $\wedge$ be the max operator.

Note: the order on the lattice is opposite to the order on $\mathbb{R}$: $(\leq) = (\geq_{\mathbb{R}})$.

In the literature there is a need to prove that evaluation preserve sensitivity, in this framework no special preservation proof is needed: any system is type-preserving for any modality ringoid, thus any assignment of types to metrics will do.

Information-Flow Security

One application of type systems is to ensure that certain parts of a program do not have access to private (high security) information.

The principal property of such systems is that the output of a program does not depend on secret inputs.

This property holds for Λ^p, if we consider that any modality p above 1 in the lattice is secret.

Note: The addition is relegated to play the same role as the meet: $(+) = (\wedge)$.
If we need a variable in two parts of a term, we mus assume the worst and require the most public level given by the meet.

Note: The multiplication acts as the join $(\vee)$ of the lattice (dual to the meet).

$$\frac{\vdash t : {}^{p}A \to B \qquad x : {}^{q}X \vdash u : A}{x : {}^{p \vee q}X \vdash t\, {}^{p}u : B}$$