## *Object Foundations*

Smalltalk

KML

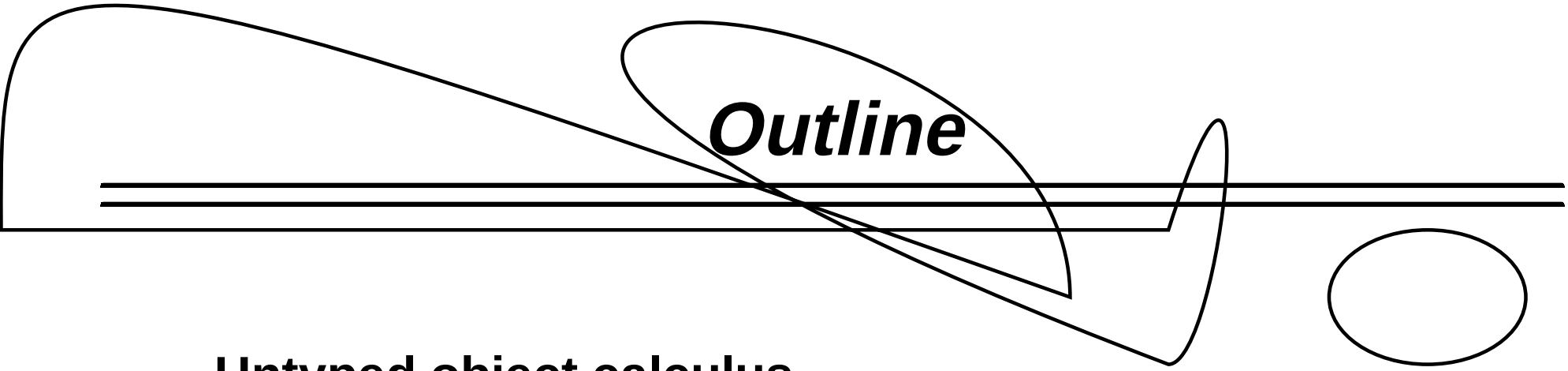
MOO

Java

CLOS

Eiffel

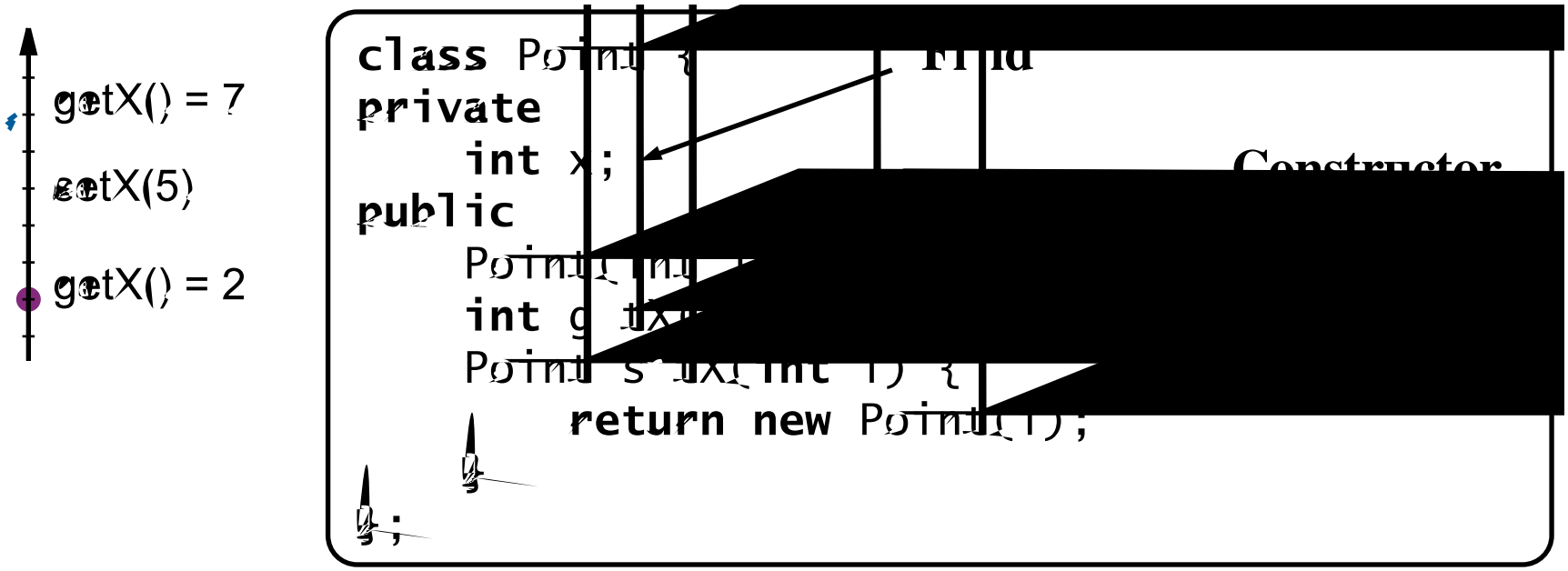
Flavors




- **Untyped object calculus**
- **Type systems & interpretations**
- **Formal interpretation**
- **Advanced object calculi**

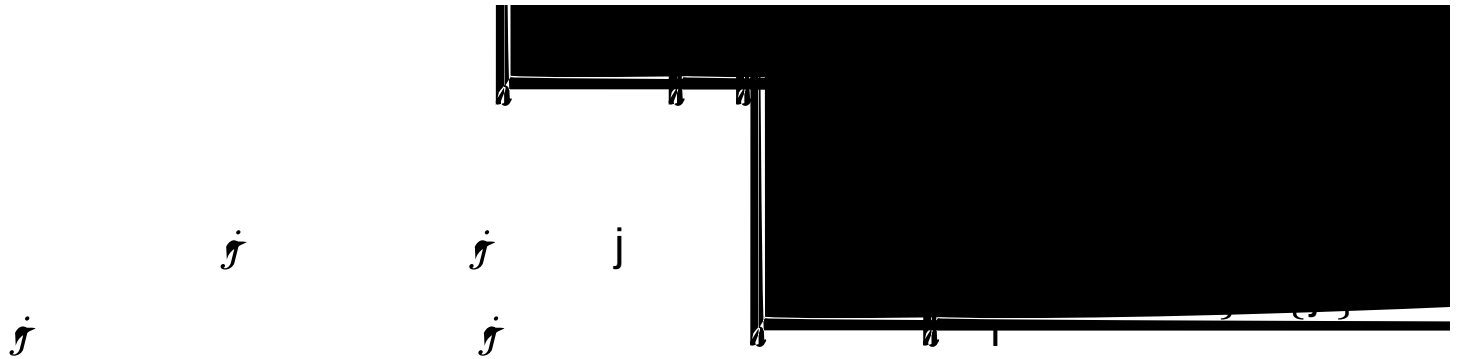
# Object Oriented Programming

Example class



# *Untyped object calculus*

(Cardelli)

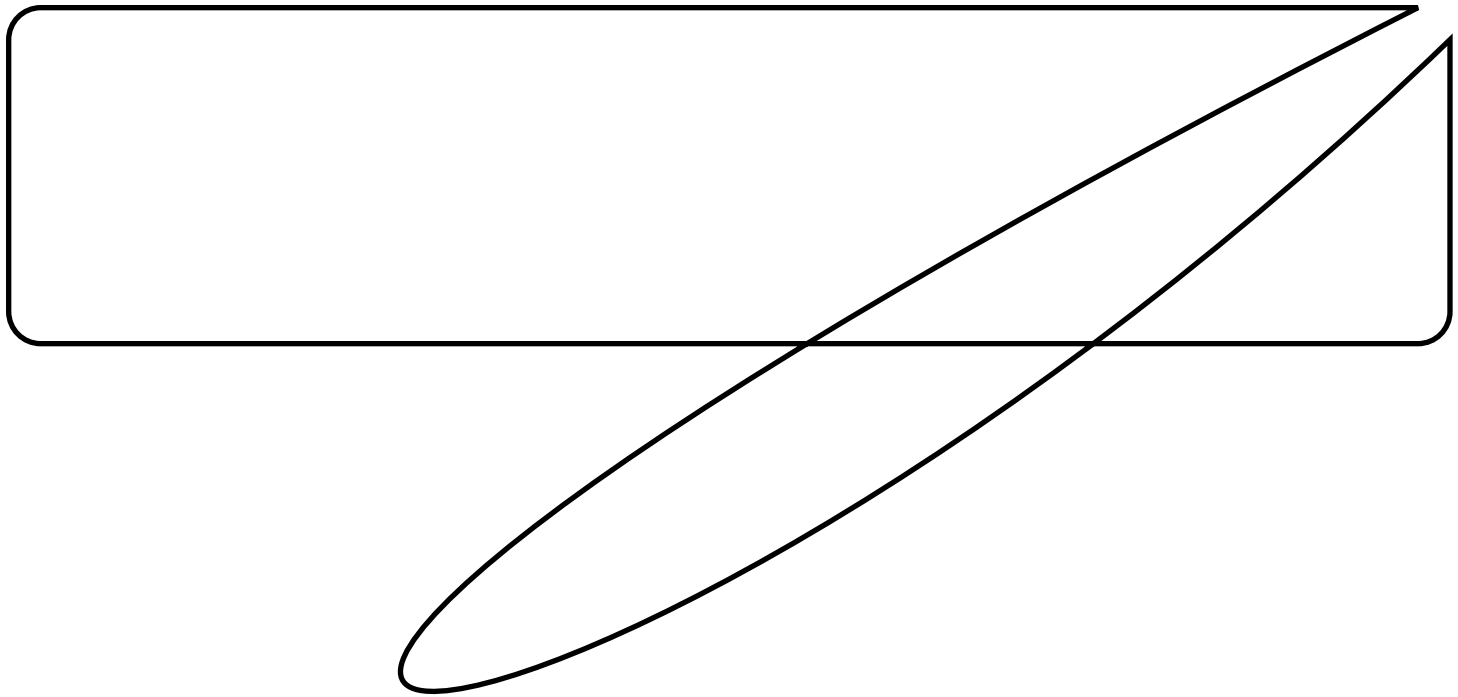


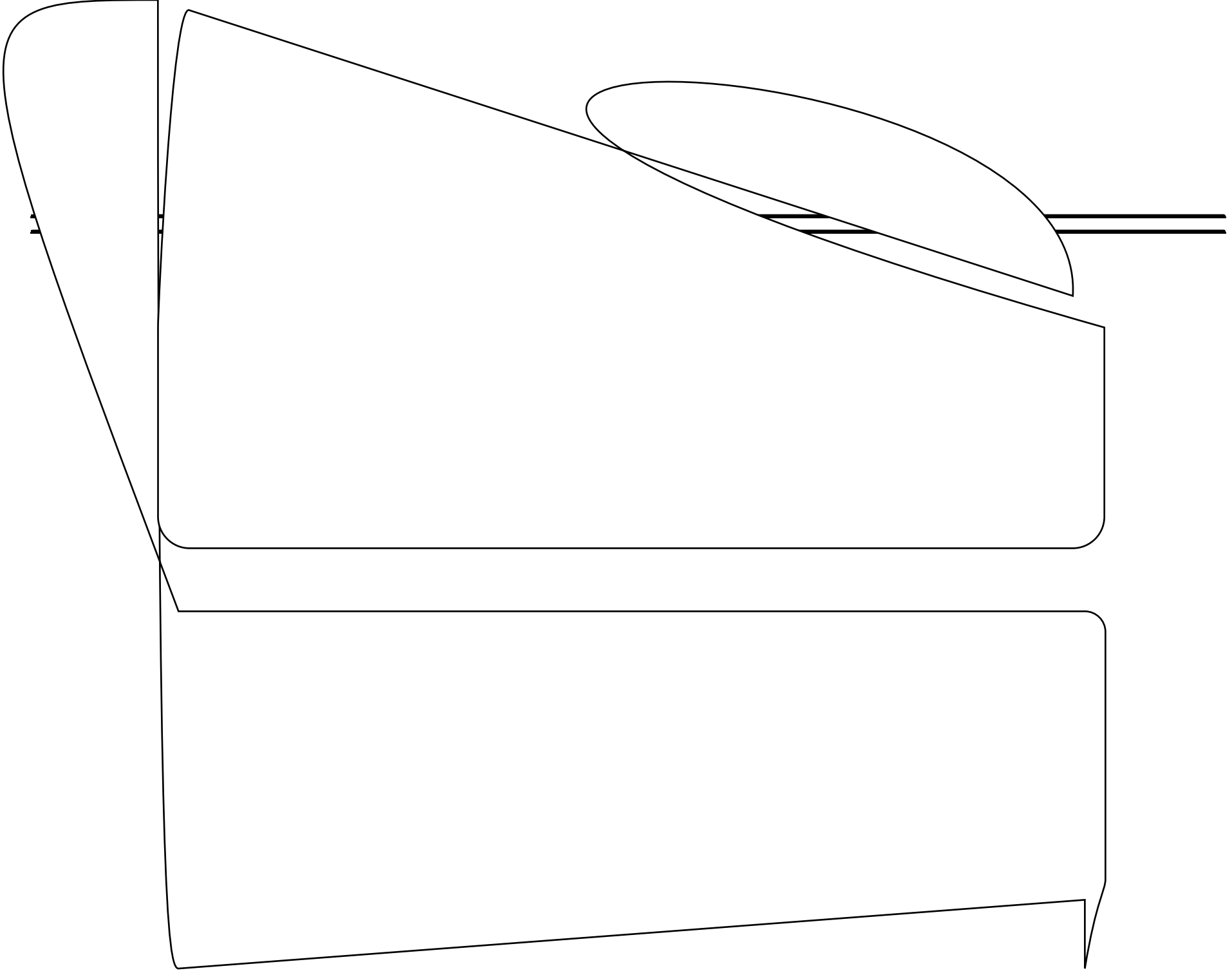
# Examples

- **1D point at location 5**

$$x = [\ getX5, moveX \leftarrow \lambda i.\ x.getX + i\ ]\ i$$

**Green colored point at location 5**





# Typed Object Calculi

$$o = [l_i = \varsigma o.\ m_i^{\ i\in\{1\ldots n\}}\ ]$$
$$o \in Obj(X.[l_i : X \to \quad _i(X) \qquad ])$$

$$\mathrm{Let}\ I(X) = \{l_i : \quad (X) \quad \}$$

$$\begin{array}{rcll}
OR(I) & = & \mu( & (X \\
OE(I) & = & \exists Y. & Y \times (Y \to I(Y)) \quad \text{(Pierce)} \\
ORE(I) & = & \mu(X. \ \exists & \times (Y \to I(Y \\
ORBE(I) & = & \mu(X. \ \exists Y &
\end{array}$$

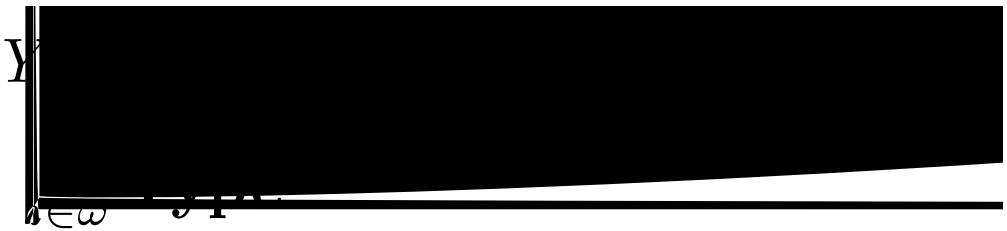
# *Advantages of formal interpretation*

Type theoretic

- **Many models**
  - PER Models
  - Set theory
- **Expressive typing**
- **Increase the communication between OO and formal commuities**
  - Better programming guag
  - Better formal systems

# Problems with type theory

$$T = \mu(X.\exists Y \leq X.\, Y(Y \to I(Y$$



More formally, let $\mathbf{Type} = \bigcup_{n \in \omega}$ Type$_n$

$\omega \quad \omega \quad \omega \quad \omega \quad \in \mathbf{Type}$

$\to \mathbf{e}$

# Operational Interpretation

- **Records**

$$[l_i = \varsigma x_i.m_i \ ^{i\in\{1\ldots n\}}] \quad \equiv \quad \{l_i = \lambda x_i.m_i \ ^{i\in\{1\ldots n\}}\}$$

$$o.l_j$$

- **Functions**

# Type Interpretation

- **Point example**
- **Construct methods inductively**

$$Point = Obj(X.[getX : X \to \mathbb{Z}; moveX : X \to \mathbb{Z} \to X])$$

$$P_0 = [getX : \mathbf{Top}$$

# *Restrictions*

- **Restrictions**
  - Total recursive objects
  - Well-founded method order
  - Method types are monotonic in Self type
- **These restrictions enforce a complete theory**
- **Interpretation is easier in a type theory of partial functions**

# Subsumption

- **Problem**
  - 
  - moveX returns a ColorPoint?
- Solution

# *Override*

- **Problem**
  - y
- **Solution**

# *Subobject (Cardelli, Abadi)*

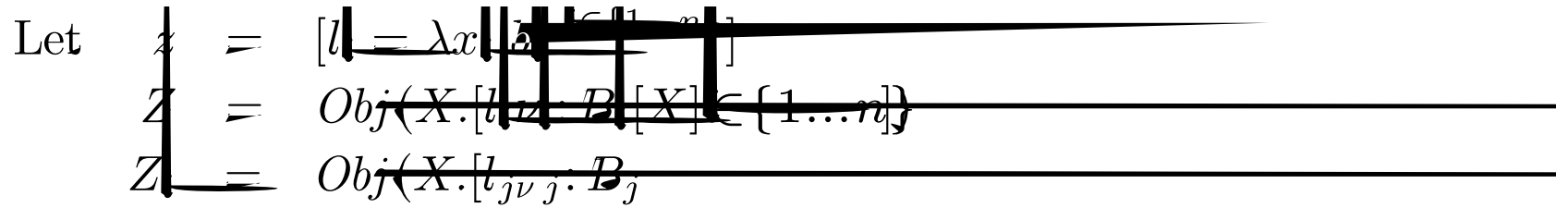
o i 2 f 1:::n }
i i

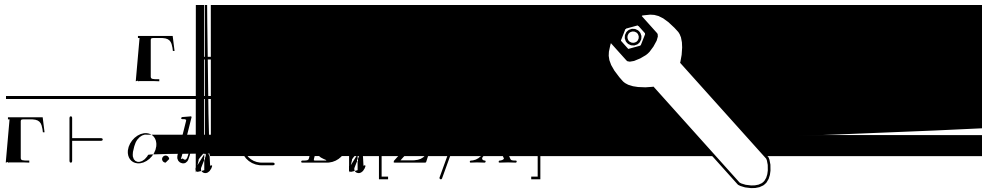
±

# General Constructh1

$$T = Obj(X:[\Pi_i^{o} : X ! \quad M_i[X]$$

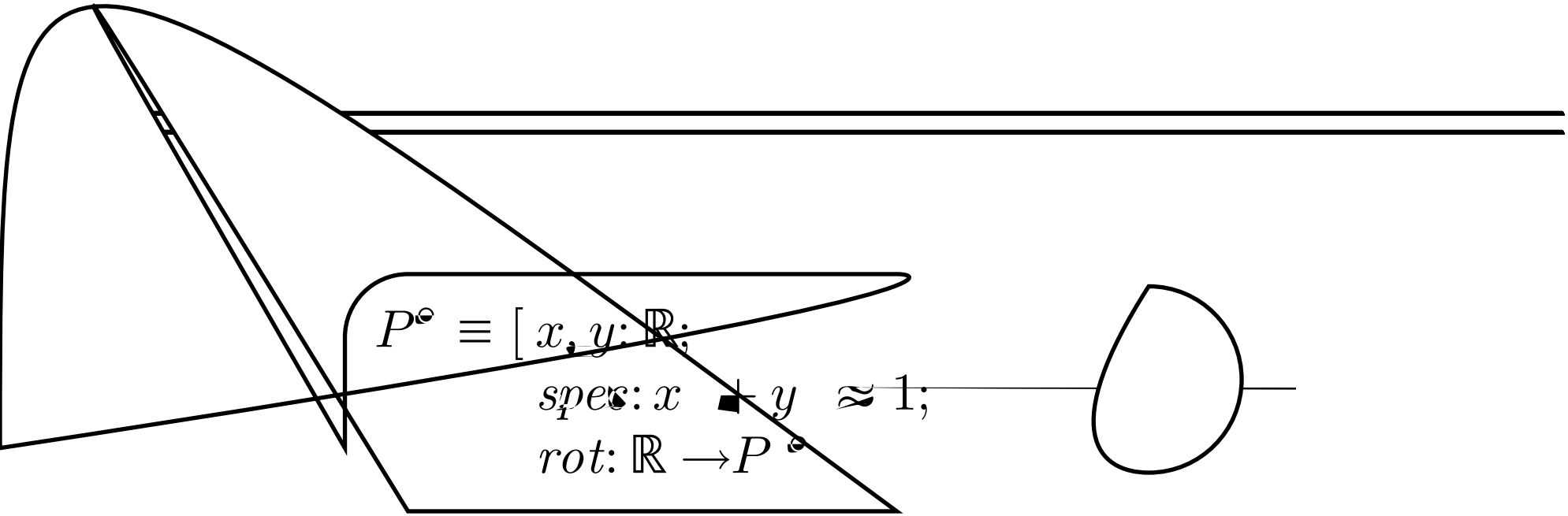
# Soundness


Let
$x = [l_i = \lambda x{:}B_i.\, b_i \mid i \in [1..n]]$
$Z = Obj(X.[l_i : B_i[X] \mid i \in \{1..n\}])$
$Z = Obj(X.[l_{j\nu\,j} : B_j$



# *Intuition*

- **An object is collection of methods that are polymorphic over subobjects**
- **This is the important factor in inheritance**
- **Polymorphism must be constrained only because of update**


$P^{\bullet} \equiv [\, x, y\colon \mathbb{R};$
$spec\colon x \perp y \approx 1;$
$rot\colon \mathbb{R} \to P^{\bullet}$

# *My work*

Modular theories and proof assistants