What can logic do for AI?

David McAllester
TTI-Chicago

Motivating Type Theory

Meta-Mathematics: Type Theory as Cognitive Science

Mathematics exists as a human social enterprise.

Modern mathematicians tend to be untrained in mathematical logic.

Can we study and understand how mathematicians (untrained in logic) actually think?

Can we model naturally occurring mathematical mentalese?

The Grammar of Mathematics (of Mentalese)

Although English sentences have grammar, English speakers must be explicitly trained in grammatical analysis.

In the same way, mathematics has a grammar — a notion of well formedness — which is used without explicit formalization.

Theories of the grammar of mathematics (the grammar of mathematical mentalese) should be viewed as part of cognitive science or artificial intelligence.

Cognitive Phenomenon I

The Identity of Indiscernibles (Leibniz)

We tend to identify isomorphic objects. Consider

- The complete graph K_n.
- The topological sphere S^n.
- The Poincaré conjecture that S^3 is the only simply connected compact three manifold.

It seems that every well-formed (grammatical) concept (every well-formed class) has an associated notion of isomorphism.

Cognitive Phenomenon II

Cryptomorphism (Birkoff, Rota)

There are different but equivalent definitions of a group:

- A group can be defined as a pair of set and a binary operation such that an identity element and an inverse operation exist.
- A group can be defined as a four-tuple of a set, group operation, identity element an inverse operation.

These two definitions are "equivalent" or "cryptomorphic". The term "cryptomorphism" is due to Birkoff and was promoted by Rota in the context of matroids.

Can we formally define the cryptomoprhism equivalence relation on definitions?

Cognitive Phenomenon III

Naturality and Voldemort's Theorem

It is intuitively clear that there is no distinguished (or natural) point on a geometric circle.

Similarly, there is no distinguished node of the complete graph K_5, no distinguished basis for a vector space, and no distinguished isomorphism between a finite dimensional vector space and its dual.

In such cases objects exist which cannot be named — there are points on the circle but no particular point can be named.

Can we prove that these objects cannot be named by grammatical expressions?

Concepts and Grammar

Type theory is the study of grammaticality in mathematics.

A type theory defines a space of concepts (types) which govern grammaticality.

We seek a notion of concept and of grammaticality that is as close as possible to naturally occurring mathematical mentalese.

Motivating Compositional Semantics

Putting Formal Expressions in Correspondence with Mentalese:

For two sets s and w we will write $s \to w$ for the set of functions from s to w.

$$\mathcal{V}_\Gamma [\![\sigma \to \tau]\!] \rho = \mathcal{V}_\Gamma [\![\sigma]\!] \rho \to \mathcal{V}_\Gamma [\![\tau]\!] \rho$$

$$\mathcal{V}_\Gamma [\![f(e)]\!] \rho = (\mathcal{V}_\Gamma [\![f]\!] \rho)(\mathcal{V}_\Gamma [\![e]\!] \rho)$$

$$\mathcal{V}_\Gamma [\![\Phi \vee \Psi]\!] \rho = \mathcal{V}_\Gamma [\![\Phi]\!] \rho \vee \mathcal{V}_\Gamma [\![\Psi]\!] \rho$$

$$\mathcal{V}_\Gamma [\![\neg\Phi]\!] \rho = \neg\mathcal{V}_\Gamma [\![\Phi]\!] \rho$$

$$\mathcal{V}_\Gamma [\![\forall x\!:\!\tau\ \Phi[x]]\!] \rho = \textbf{True} \text{ iff } \begin{cases} \text{for every } u \in \mathcal{V}_\Gamma [\![\tau]\!] \rho \\ \text{we have } \mathcal{V}_{\Gamma;\, x:\tau} [\![\Phi[x]]\!] \rho[x := u] = \textbf{True} \end{cases}$$

Platonism

[Markus Maurer]

Mathematical practice (and thought) is Platonic.

Platonism is simply using one's own native mentalese.

The formulas of mentalese have variables ranging over "the objects themselves".

Morphoid Type Theory

variables, pairs	x	(e_1, e_2)	$\pi_1(e)$	$\pi_2(e)$
functions	$\lambda x\!:\!\sigma\ e[x]$	$f(e)$		
Booleans	$P(e)$	$e_1 \doteq e_2$	$e_1 =_\sigma e_2$	
	$\neg\Phi$	$\Phi_1 \vee \Phi_2$	$\forall x\!:\!\sigma\ \Phi[x]$	
types	**Bool**	**Set**	**Class**	**Kind**
	$\Sigma_{x:\sigma}\ \tau[x]$	$\Pi_{x:\sigma}\ \tau[x]$	$S_{x:\sigma}\ \Phi[x]$	
contexts	ϵ	$\Gamma; x\!:\!\tau$	$\Gamma; \Phi$	
sequents	$\Gamma \vdash e :: \sigma$	$\Gamma \vdash e\!:\!\sigma$	$\Gamma \vdash \Phi$	

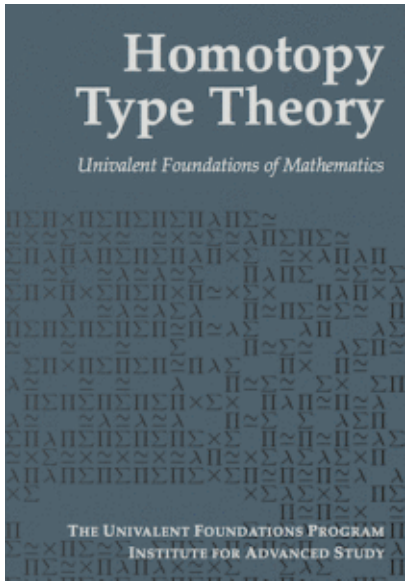
Martin Löff Type Theory

Per Martin Löff, An intuitionistic theory of types, 1975.

Martin Löff type theory (MLTT) dominates type-theoretic mathematical foundations today.

MLTT carries the baggage of constructivism and propositions as types — baggage that blocks any direct correspondence with mentalese.

Homotopy Type Theory: Equality as Isomorphism in MLTT



Homotopy Type Theory, 2013

Advocates "informal" mathematics based on MLTT and univalence.

It took me a long time to realize that this book does not define the meaning of the notation.

Two Key Type Expressions

$$\mathcal{V}_\Gamma \, [\![\Sigma_{x:\sigma} \, \tau[x]]\!] \, \rho \; = \; \{(a, b), \; a \in \mathcal{V}_\Gamma \, [\![\sigma]\!] \, \rho, \; b \in \mathcal{V}_{\Gamma;\, x:\sigma} \, [\![\tau[x]]\!] \, \rho[x := a]\}$$

$$\mathcal{V}_\Gamma \, [\![S_{x:\sigma} \, \Phi[x]]\!] \, \rho \; = \; \{a \in \mathcal{V}_\Gamma \, [\![\sigma]\!] \, \rho, \; \mathcal{V}_{\Gamma;\, x:\sigma} \, [\![\Phi[x]]\!] \, \rho[x := a] = \mathbf{True}\}$$

Examples of Σ-Types and Subtypes

The type of directed graphs can be written as

$$\mathbf{DiGraph} \equiv \Sigma_{\mathcal{N}:\mathbf{Set}}\, \mathcal{N} \times \mathcal{N} \to \mathbf{Bool}$$

$$\mathbf{HyperGraph} \equiv \Sigma_{\alpha:\mathbf{Set}}\ \ (\alpha \to \mathbf{Bool}) \to \mathbf{Bool}$$

$$\mathbf{TOP} \equiv S_{X:\mathbf{HyperGraph}}\ \Psi[X]$$

$$\vdash \mathbf{TOP}:\mathbf{Class}$$

"Internalizing" Isomorphism

We define a **simple type** over a type variable α by the following grammar

$$\tau \ ::= \sigma \text{ not containing } \alpha \mid \alpha \mid \ \mathbf{Pair}(\tau_1, \tau_2) \ \mid \tau_1 \to \tau_2$$

A simple Σ-type is a type of the form $\Sigma_{\alpha:\mathbf{Set}}\ \tau[\alpha]$ where $\tau[\alpha]$ is a simple type over α.

For a simple Σ-type we have that (s, a) is isomorphic to (s', a') if there exists a bijection from s to s' that carries a to a'.

For a simple type $\tau[\alpha]$ the carrying relation between $\tau[s]$ and $\tau[s']$ is easily defined by structural induction on $\tau[\alpha]$.

Bag of words example

Let V be a vocabulary of words (let V be a set).

Define a totally ordered set (TOS) to be a pair $(S, \leq)$ where S is a set and $\leq$ is a total order on S.

Define a document over vocabulary V to be a pair of a totally ordered set $(S, \leq)$ and a function $f : S \to V$.

$$\text{DOC} \equiv \Sigma_{I:\text{TOS}} \ \pi_1(I) \to V.$$

The bag of words abstraction of a document (I, f) is the isomorphism class of $(\pi_1(I), f)$ in the class $\Sigma_{\alpha:\mathbf{Set}} \ \alpha \to V$.

Substitution of Isomorphics

$$\frac{\Gamma; x:\sigma \vdash \Phi[x]:\mathrm{Bool} \quad \Gamma \vdash u =_\sigma w}{\Gamma \vdash \Phi[u] \Leftrightarrow \Phi[w]}$$

or more generally

$$\frac{\Gamma; x:\sigma \vdash e[x]:\tau \quad x \text{ not free in } \tau \quad \Gamma \vdash u =_\sigma w}{\Gamma \vdash e[u] =_\tau e[w]}$$

The Hard Part

$$\mathcal{V}_\Gamma [\![u \ =_\sigma \ w]\!] \, \rho \quad = \quad \mathcal{V}_\Gamma [\![u]\!] \, \rho \ =_{\mathcal{V}_\Gamma [\![\sigma]\!] \rho} \ \mathcal{V}_\Gamma [\![w]\!] \, \rho$$

What does $a =_\sigma b$ mean for an arbitrary class σ?

$\mathcal{V}_\Gamma [\![\mathbf{Class}]\!] \, \rho =?$ what is a class?

In morphoid type theory a class is a collection. The class denoted by a closed class expression can be assigned groupoid structure. But in general (for open class expressions) a class is a collection that can be assigned "morphoid" structure. This is a long story.

Modeling General Natural Language

Logic in Support of General Semantics

Paul Manafort is said to have proposed a strategy to nullify anti-Russian opposition across former Soviet republics a decade ago.

Manifort:person

Proposal37:proposal proposal $\subseteq$ event

Proposal37.agent = Manifort

Proposal37.object = Strategy52

Proposal37.time $\subseteq$ a decage ago.

Proposal37.recipient = ?

Strategy52:strategy

Strategy52.purpose = nullify Opposition73

$\vdots$

Soft Inference Rules

If x proposed y to z then x wanted z to accept y.

If x is nullified, and x.purpose $= y$, then y is prevented.

Writing down an adequate set of rules is hopeless.

But maybe the rules can be learned.

But what is an appropriate Neural Architecture and Training Task?

Seeking a Universal Neural Architecture

There are many models of computation (programming languages and/or architectures).

They are all Turing universal (the Turing tar pit).

However, they are not all equal.

Is there a distinguished "deep logic" architecture?

Bottom-up Logic Programming

Consider a database D and a set of inference rules R.

Let $R(D)$ be the assertions derivable from D using rules in R.

Inference rules naturally express dynamic programming algorithms.

A rule is "local" if it does not introduce new entities.

Theorem: Local rules "capture" the complexity class P — we have $\mathcal{L} \in P$ if and only if there exists R such that

$$\text{Accept} \in R(\text{Input}(t)) \quad \text{iff} \quad t \in \mathcal{L}.$$

Deep Logic Programming

I will define a neural database to be a graph D such that for each node n of G we have an **entity embedding** $e(n)$ and for each directed edge (n, m) of G we have a relationship vector $\Phi(n, m)$.

A set of inference rules then defines a graph transformation.

We consider rules stated in terms of predicate symbols.

listening-to(x, y), said(y, P) $\Rightarrow$ heard(x,P).

$$\Phi(x, P) \texttt{ += } \alpha\ e(\text{heard})$$

$$\alpha \ = \ (\Phi(x, y) \cdot e(\text{listening-to}))\ (\Phi(y, P) \cdot e(\text{said}))$$

Linguistic Reference

In language comprehension we can take each word occurrence to be an entity (the referent of the phrase headed by that word occurrence).

Coreference can be treated with congruence-closure-like deep rules — just part of the same "bottom-up" deep logic architecture.

[Logical Algorithms, Ganzinger and McAllester, ICLP, 2002]

END