Systems

Nigel Redding

May 27, 2026

1 Categories and Software

Imagine a large software system, where types are modeled using google protocol-buffers. Each type in the system can be translated into lambda calculus, in a similar way that we can use lambda calculus to model non-negative integers. We construct a category $\mathcal{D}$ which contains these types, and an arrow between type $\mathbf{t}_1$ and $\mathbf{t}_2$ is a computable function $\mathbf{t}_1 \to \mathbf{t}_2$. The set (or class) $\mathcal{T}$ of objects in this category not only contains the types given by the protocol-buffers, but also arrows between them, and so on. So if $\mathbf{t}_1$ and $\mathbf{t}_2$ are types, then so is $\mathbf{t}_1 \to \mathbf{t}_2$, which we view as the set of computable functions from $\mathbf{t}_1$ to $\mathbf{t}_2$. We can also denote it by $\hom(\mathbf{t}_1, \mathbf{t}_2)$. [wip]

2 Why LLMs are so powerful

Once we view systems through the lens of language theory and AI, we get a better view of how to build AI tools to assist with system design and implementation. The basic argument is as follows. We can model our system with a formal language. We can assume that we can translate between our formal language and a natural language in a reasonably efficient way. We can encode the notion of "truth" into our system, by training it on sentences generated from the formal language. The underlying logic of the formal language gets embedded into natural language. This explains why LLMs are so powerful, and can effectively reason about systems.

3 Languages and Models

Recall the definition of a **formal language**. Let Σ be a nonempty set, which we call the alphabet. Let ε be a distinguished symbol which we call the *empty string*. Define a word w to be a finite sequence of elements of Σ, or $w = \varepsilon$. We define *concatenation* of words as follows. We define $\varepsilon w = w\varepsilon = w$ for all words w. Let w_1 and w_2 be non-empty words. Then we define

$$w_1 * w_2 = x_1 x_2 \cdots x_M y_1 y_2 \cdots y_N \qquad w_1 = x_1 \cdots x_M \quad w_2 = y_1 \cdots y_N.$$

We call this structure the *Kleene-star* and denote it by Σ^*. The reader may also know that this is an instance of a *free monoid*. We define a *language* L to be a non-empty subset of Σ^*. We define a *model* of L to be a pair (v, D) where D is a non-empty set and $v : L \to D$ is a function.

Let us consider propositional logic, which is the language defined by a basic set of propositions **Prop**. Consider the connectives

$$\wedge, \vee, \neg, \to$$

each of which can be defined through an obvious truth table. We define the language CPL by inductively building sentences with the connectives. This is called *classical propositional logic*. As a choice of model, we could define all the propositions in **Prop** to be true (i.e. **Prop** is our basic set of facts). Alternatively, its possible that in a given situation, or *world*, some of the propositions will be true and others will be false. We explore this in a later section.

Recall that we have a "turnstile" symbol $\vdash$, which is not part of propositional logic. Instead, it belongs to a meta-language. It works as follows. If P is a set of premises, and one can derive Q from P (using logic and direct manipulation), then we say

$$P \vdash Q.$$

If we have no premises, then

$$\vdash Q$$

means that Q follows from the axioms of the system with **no assumptions**.

4 A Simple Model for Legal Contracts

We can extend CPL to describe things like: obligations, time-relations, etc.. In this section, we extend CPL to get another language $DCPL$, which is the smallest language containing L and the set

$$\{\mathcal{O}(p) : p \in L\}$$

and we pronounce $\mathcal{O}(p)$ by saying that *there is an obligation that p*. We call $DCPL$ **deontic logic**. We define the *permission* and *forbidden* operators by

$$\mathcal{P}p := \neg\mathcal{O}\neg p$$

and

$$\mathcal{F}p := \mathcal{O}\neg p$$

respectively.

We have the language defined. To restrict the set of models to sensible ones, we impose the following axioms, which we recall are not axioms for the language, but merely impositions on any model.

1. **K** (distributive) :
$$\mathcal{O}(p \to q) \to (\mathcal{O}(p) \to \mathcal{O}(q))$$
2. **D** (serial) :
$$\mathcal{O}p \to \neg\mathcal{O}\neg p$$
3. **N** (necessity) :
$$\vdash p \to \vdash \mathcal{O}p$$

With DCPL defined, we move on to defining a model of the language.

Recall that for a set X, 2^X denotes its power set, i.e. the set of subsets of X. A Kripke model $\mathcal{M}$ consists of the following three pieces of data

- A set of worlds W
- A relation $R \subset W \times W$
- A function (called a **valuation**)

$$V : \mathbf{Prop} \to 2^W \qquad V(p) = \{w \in W : p \text{ is true in } w\}.$$

For a world w and a proposition p, we say w **models** p, denoted by $w \models p$ if $w \in V(p)$. We extend the definition of $\models$ on atomic propositions as follows:

$$\begin{aligned} w \models \neg p &\iff \neg(w \models p) \\ w \models (p \wedge q) &\iff (w \models p) \wedge (w \models q) \\ w \models (p \vee q) &\iff (w \models p) \vee (w \models q) \\ w \models (p \to q) &\iff (w \models p) \to (w \models q) \end{aligned}$$

We now define it on SDL sentences. We say

$$w \models \mathcal{O}p \iff \forall w \in W : vRw \to v \models p$$

and

$$w \models \mathcal{P}p \iff \exists v \in W : vRw \wedge v \models p.$$

DCPL suffers from the problem of not being able to handle the notion of *conditional obligation*, and hence fails to model the legal notion of damages. Indeed, if we want to say "if you break the vase, you are obligated to pay for it", then we can model it formally, where the variables are

- p : "you break the vase"
- q : "you pay for the vase"

so we are tempted to model this proposition by

$$p \to \mathcal{O}(q).$$

This is a bad idea, as we recall that $p \to \mathcal{O}(q)$ really means $\neg p \vee q$, i.e. you do not break the vase, or the you are obligated to pay for the vase, or both. We remedy this by extending the language DCPL to a new language DDCPL, *dyadic deontic classical propositional logic*, which we outline below.

Recall the notion of **order** in mathematics, and let us consider orders on W. We say a relation $\preceq \subset W \times W$ is a **partial order** if it is both reflexive and transitive. We say $\preceq$ is a **total order** if it is total, i.e. for any $w_1, w_2 \in W$, $w_1 \preceq w_2$ or $w_2 \preceq w_1$, and antisymmetric, i.e. if $w_1 \preceq w_2$ and $w_2 \preceq w_1$ then $w_1 = w_2$.

This distinction is important. Let $\preceq$ be a partial order on W. Then if $U \subset W$, then we say $x \in U$ is **minimal** if $x \preceq u$ for all $u \in U$. If the order is total, then this element is unique. If it is not total, then it is possibly non-unique. We denote the set of minima by

$$\min_{\preceq} U.$$

Now assume that for each world $w \in W$ we have a partial order $\preceq_w \subset W \times W$ relative to w. Then we consider

$$[p] = \{w \in W : w \models p\}$$

and we define

$$w \models \mathcal{O}(p|q) \iff \min_{\preceq_w}[q] \subset [p].$$

We say $(M, W, \preceq, V)$ is a **dyadic deontic model**.

We are now ready to define a simple class of legal contract. Let $(M, W, \preceq, V)$ be a dyadic deontic model. We define a **contract** as

$$\mathcal{C} = \{\mathcal{O}(p_1|q_1), \ldots, \mathcal{O}(p_n|q_n)\}$$

and say $\mathcal{C}$ is **satisfied** with respect to $w \in W$ if $M, w \models \phi$ for all $\phi \in \mathcal{C}$. We denote this by

$$M, w \models \mathcal{C}.$$