(CR15) CATEGORY THEORY FOR COMPUTER SCIENTISTS: LECTURE 1

12 SEPTEMBER 2024 — L. T. D. NGUYỄN

A computer science motivation: Programming language semantics

Let us briefly discuss one of the uses of categories for computer science — though by no means the only one: the coalgebra course (CR17) provides another.

To formally study programs, one needs to rigorously define their meaning. The definition should ideally be "high level" and independent of, say, any specific instruction set architecture. The approach known as *denotational semantics* interprets expressions in programs as mathematical objects: the meaning of a function `f` of type `(int -> int) -> int` would naively be a map $[\![\mathtt{f}]\!] \colon \mathbb{Z}^{\mathbb{Z}} \to \mathbb{Z}$.

This naive idea only works in toy cases: it does not "scale up" to complex features of programming languages. In order to interpret loops and recursion by "taking the limit as $N \to \infty$ of N iterations", the first denotational semantics that appeared[1] in the late 1960s / early 1970s used topology:

- the types `int`, `int -> int`, etc. are interpreted as (weird) "spaces", which are sets of points endowed with extra structure;
- $[\![\mathtt{f}]\!] \colon [\![\mathtt{int}]\!] \to [\![\mathtt{int \text{ -> } int}]\!]$ should be a continous map, i.e. a map between the sets of points that "preserves" the extra structure.

For a while, denotational semantics relied on this paradigm of "structured sets" with variations on the structure. But in later models of programming languages (quantitative semantics, game semantics, ...), the interpretation $[\![\mathtt{f}]\!]$ is not even a set-theoretic map! This parallels the 20th century development (cf. [Cor04]) of structural mathematics, starting from structured sets (vector spaces, groups, etc.) and eventually leading to the invention of category theory [EML45] when the need for a more abstract setting arose in algebra and topology.

Thus, category theory provides a way to axiomatize "what is a semantics of some given language" that is general enough to cover this wide variety of concrete semantics. In return, categorical semantics has inspired features of programming languages — either theoretical or actually implemented. The most famous example is that of monads in functional programming [Pet18] (i.e. burritos [Yor09]) which came from a semantic account of side effects (such as printing "Hello world!").

Reminder: monoids as an example of algebraic structures

Many of the usual "basic" examples of categories involve algebraic structures. We avoid advanced examples here, but recall a basic one, which plays a major role in computer science (for instance in automata theory).

Definition. A *monoid* is $(M, \cdot, e)$ where:

- M is a set
- $\cdot$ is a binary operation $M \times M \to M$
- e is an element of M (the "unit")

such that

[1] For a history of programming language semantics, see [Ast19].

associativity: $\forall x, y, z \in M,\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$
unitality: $\forall x \in M,\ x \cdot e = e \cdot x = x$

There are many **examples** (below, X is a set):

- $(\mathbb{N}, +, 0)$, $(\mathbb{Z}, +, 0)$, $(\mathbb{R}, +, 0)$
- $(\mathbb{N}, \times, 1)$, $(\mathbb{Z}, \times, 1)$, $(\mathbb{R}, \times, 1)$
- $(X^*, \cdot, [])$ where X^* is the set of finite lists with elements in X, the operation $\cdot$ is list concatenation and $[]$ is the empty list
- the set of all functions $X \to X$ with function composition and the identity function id_X
- the set of all bijections $X \xrightarrow{\sim} X$ with composition and the identity

Remark. *Abuse of notation:* we may write M instead of $(M, \cdot, e)$ when there is no ambiguity on the operation and the unit (so, for instance, for "the monoid X^*").

Remark. Associativity allows us to write $x_1 \cdot \ldots \cdot x_n$ without parentheses — or even $x_1 \ldots x_n$ — when $x_1, \ldots, x_n$ are elements of some monoid M. Typically in $(\mathbb{N}, +, 0)$ we can write $1 + 3 + 42 + 7$ and the meaning is unambiguous.

Definition. A *homomorphism* of monoids from $(M, \cdot_M, e_M)$ to $(M', \cdot_{M'}, e_{M'})$ is a map $h \colon M \to M'$ such that

- $\forall x, y \in M,\ h(x \cdot_M y) = h(x) \cdot_{M'} h(y)$
- $h(e_M) = e_{M'}$

Again, many examples:

- the inclusion map $x \in \mathbb{N} \mapsto x \in \mathbb{R}$ is both a homomorphism from $(\mathbb{N}, +, 0)$ to $(\mathbb{R}, +, 0)$ and a homomorphism from $(\mathbb{N}, \times, 1)$ to $(\mathbb{R}, \times, 1)$
- $x \mapsto (-2)^x$ is a homomorphism from $(\mathbb{N}, +, 0)$ to $(\mathbb{Z}, \times, 1)$
- taking the length of a list defines a homomorphism from X^* to $(\mathbb{N}, +, 0)$

Proposition (closure under composition)**.** *The composition of a homomorphism from M to M' with a homomorphism from M' to M'' is itself a homomorphism from M to M''.*

Furthermore, since id_M is a homomorphism from M to M, this implies that the homomorphisms from M to M — also called the *endomorphisms* of M — form a monoid with function composition!

Definition. A homomorphism $f \colon M \to M'$ is an *isomorphism* when there exists a homomorphism $g \colon M' \to M$ such that $g \circ f = \mathrm{id}_M$ and $f \circ g = \mathrm{id}_{M'}$.

Necessarily, if this is the case, f must be a bijection and g must be its *inverse:* $g = f^{-1}$. As an example:

- For lists over a singleton set $\{a\}$, the length homomorphism $\{a\}^* \to \mathbb{N}$ is an isomorphism. Its inverse is $n \mapsto [a, \ldots, a]$ (n times).

Actually, when f is a bijective homomorphism, f^{-1} is automatically a homomorphism too; in other words:

Proposition. *A homomorphism $f \colon M \to M'$ is an isomorphism* if and only if *it is a bijection.*

But this does not always work with all algebraic structures; the "right" definition of isomorphism is the first one.

We say that M and M' are *isomorphic* when some isomorphism from M to M' exists. This means that they "are the same" with respect to the monoid structure. Notation: $M \cong M'$.

An example of universal property: the free monoid

It is often said that "X^* is the free monoid over the set X". This refers to the following property. Let $\iota\colon x \in X \mapsto [x] \in X^*$.

Theorem (Universal property of X^*)**.** *Let X be a set. For any monoid $(M, \cdot, e)$ and map $f\colon X \to M$, there exists a unique homomorphism $h\colon X^* \to M$ such that $f = h \circ \iota$.*

Proof. First, note that $f = h \circ \iota$ means that $\forall x \in X,\ f(x) = h([x])$.

For uniqueness, assume that h satisfies this equation. Then $h([]) = e$ because a homomorphism must preserve the units, and for $n \in \mathbb{N}$,

$$\begin{aligned} h([x_1, \ldots, x_{n+1}]) &= h([x_1, \ldots, x_n] \cdot [x_{n+1}]) \\ &= h([x_1, \ldots, x_n]) \cdot h([x_{n+1}]) \\ &= h([x_1, \ldots, x_n]) \cdot f(x_{n+1}) \end{aligned}$$

By induction we have $h([x_1, \ldots, x_n]) = f(x_1) \cdot \ldots \cdot f(x_n)$ and this uniquely defines f because every element of X is of this form for some $n \in \mathbb{N}$.

For existence, we can check that the above equation always defines a homomorphism h such that $f = h \circ \iota$. □

This property is usually drawn as the following *commutative diagram:*

$$\begin{tikzcd} X \arrow[r, "\iota"] \arrow[dr, "f"'] & X^* \arrow[d, dashed, "\exists! h \text{ homomorphism}"] \\ & M \end{tikzcd}$$

By convention, we use "$\to$" arrows to represent given data, and "$\dashrightarrow$" arrows to represent things that we then claim must exist. There are two paths from X to M in the diagram, one with a single arrow representing the map f, another with two arrows representing the composite $h \circ \iota$ ("first apply ι then apply h"). To say that the diagram *commutes* is to say that these paths denote the same map, in other words, $f = h \circ \iota$.

Let us now see, as a first illustration of the spirit of category theory, why *the universal property characterizes X^* up to isomorphism.* (Later in the course we will even say "up to unique isomorphism" but this requires some precautions.) This is how category theory identifies canonical constructions, thus serving as "essential guidance" [Mad19] in some branches of mathematics and theoretical computer science.

Proposition. *Suppose that some monoid N and some map $g\colon X \to N$ satisfy the same universal property: for any monoid M and map $f\colon X \to M$, there exists a unique homomorphism $h\colon N \to M$ such that $f = h \circ g$.*

$$\begin{tikzcd} X \arrow[r, "g"] \arrow[dr, "f"'] & N \arrow[d, dashed, "\exists! h \textit{ homomorphism}"] \\ & M \end{tikzcd}$$

Then $X^* \cong N$.

(For example, for $X = \{a\}$, we can take $N = \mathbb{N}$ and $g(a) = 1$.)

Proof. Let us prove that by reasoning only on compositions of homomorphisms using the universal property. First we apply the property for X^*, taking $M = N$ and $f = g$: there exists a unique homomorphism h_1 such that $g = h_1 \circ \iota$.

$$\begin{tikzcd}
X \arrow[r, "\iota"] \arrow[dr, "g"'] & X^* \arrow[d, dashed, "\exists! h_1 \text{ homomorphism}"] \\
& N
\end{tikzcd}$$

Then we apply the property for N, taking $M = X^*$ and $f = \iota$:

$$\begin{tikzcd}
X \arrow[r, "g"] \arrow[dr, "\iota"'] & N \arrow[d, dashed, "\exists! h_2 \text{ homomorphism}"] \\
& X^*
\end{tikzcd}$$

We would now like to show that h_1 and h_2 are inverse to each other. First let us show $h_1 \circ h_2 = \mathrm{id}_N$. The trick is to use the uniqueness part of the universal property for N. We apply with $M = N$ and $f = g$:

$$\begin{tikzcd}
X \arrow[r, "g"] \arrow[dr, "g"'] & N \arrow[d, dashed, "\exists! h \text{ homomorphism}"] \\
& N
\end{tikzcd}$$

- Since $g = \mathrm{id}_N \circ g$, the diagram commutes when h is replaced by the homomorphism id_N; *by uniqueness*, $h = \mathrm{id}_N$.
- Also, $g = h_1 \circ \iota = h_1 \circ (h_2 \circ g) = (h_1 \circ h_2) \circ g$, and $h_1 \circ h_2$ is a homomorphism (by closure under composition); so, by uniqueness again, $h = h_1 \circ h_2$.

Thus, $h_1 \circ h_2 = \mathrm{id}_N$. Symmetrically, we can use the uniqueness part of the universal property of X^* to show that $h_2 \circ h_1 = \mathrm{id}_{X^*}$. Therefore, h_1 and h_2 are mutually inverse isomorphisms of monoids. □

Remark. The computation $g = (h_1 \circ h_2) \circ g$ can be carried out diagrammatically:

$$\begin{tikzcd}
X \arrow[r, "g"] \arrow[dr, "\iota"] \arrow[ddr, "g"'] & N \arrow[d, "h_2"] \\
& X^* \arrow[d, "h_1"] \\
& N
\end{tikzcd}$$

The basic idea is that "the two small inner triangles commute (by definition of h_2 and h_1), therefore the big outer triangle commutes".

Observe that the proof of this proposition only manipulates functions, homomorphisms and their compositions, without mentioning lists or the elements of the monoids. This is category-theoretic reasoning!

Categories: basic definitions

Definition. A *category* $\mathcal{C}$ consists of:

- a collection $\mathrm{ob}(\mathcal{C})$ of *objects*
- for any two objects $A, B \in \mathrm{ob}(\mathcal{C})$, a collection $\mathcal{C}(A, B)$ of *morphisms*, which are denoted diagrammatically as

$$A \xrightarrow{f} B \quad \text{for } f \in \mathcal{C}(A, B)$$

- for any $A, B, C \in \mathrm{ob}(\mathcal{C})$, a *composition* operation

$$\circ \colon \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$$

$$\begin{tikzcd} A \arrow[rr, "g\circ f"] \arrow[dr, "f"'] & & C \\ & B \arrow[ur, "g"'] & \end{tikzcd}$$

- for each $A \in \mathrm{ob}(\mathcal{C})$, an *identity morphism* $\mathrm{id}_A \in \mathcal{C}(A, A)$

such that, for any $A, B, C, D \in \mathrm{ob}(\mathcal{C})$,

assoc.: $\forall f \in \mathcal{C}(A, B)$, $\forall g \in \mathcal{C}(B, C)$, $\forall h \in \mathcal{C}(C, D)$, $(h \circ g) \circ f = h \circ (g \circ f)$ which makes the meaning of $h \circ g \circ f$ non-ambiguous

$$\begin{tikzcd} A \arrow[r, "h\circ g\circ f"] \arrow[d, "f"'] & D \\ B \arrow[r, "g"'] & C \arrow[u, "h"'] \end{tikzcd}$$

unitality: $\forall f \in \mathcal{C}(A, B),\ f \circ \mathrm{id}_A = \mathrm{id}_B \circ f = f$

The basic examples are categories of structured sets with usual function composition and identity maps:

the "category of sets": $\mathrm{ob}(\mathbf{Set})$ = all sets, $\mathbf{Set}(A, B)$ = all maps $A \to B$
the "category of monoids": $\mathrm{ob}(\mathbf{Mon})$ = all monoids, $\mathbf{Mon}(M, N)$ = all homomorphisms $M \to N$ (this works thanks to closure under composition)

Remark. In usual foundations of mathematics (most of the time, ZFC set theory), the collection of all sets is not a set (Russell's paradox, etc.). This is why we remain vague about what a "collection" is.

Warning: The name "category of sets" is somewhat abusive: it describes only the objects, but in a category the morphisms and composition are more important than the objects! Another category with all sets as objects is

the "category of relations": $\mathrm{ob}(\mathbf{Rel})$ = all sets, $\mathbf{Rel}(A, B)$ = all subsets of $A \times B$, i.e. binary relations from A to B

Recall that the usual composition of two relations is defined as

$$R' \circ R = \{(x, z) \mid \exists y : (x, y) \in R \text{ and } (y, z) \in R'\}$$

With the identity relation $\mathrm{id}_A = \{(a, a) \mid a \in A\}$, this defines the category **Rel**. (To prove it, you should check the associativity and unitality axioms.)

Remark. A little inconsistency in naming conventions: the names **Set** and **Mon** refer to the objects of the category, but the name **Rel** refers to its morphism.

References

[Ast19] Troy Astarte. *Formalising Meaning: a History of Programming Language Semantics*. PhD thesis, Newcastle upon Tyne, 2019. URL: `http://homepages.cs.ncl.ac.uk/troy.astarte/res/pdf/TK_Astarte_Formalising_Meaning_2019.pdf`.

[Cor04] Leo Corry. *Modern Algebra and the Rise of Mathematical Structures*. Birkhäuser Verlag, 2nd edition, 2004. `doi:10.1007/978-3-0348-7917-0`.

[EML45] Samuel Eilenberg and Saunders Mac Lane. General theory of natural equivalences. *Transactions of the American Mathematical Society*, pages 231–294, 1945. `doi:10.1090/S0002-9947-1945-0013131-6`.

[Mad19] Penelope Maddy. What do we want a foundation to do? In Stefania Centrone, Deborah Kant, and Deniz Sarikaya, editors, *Reflections on the Foundations of Mathematics: Univalent Foundations, Set Theory and General Thoughts*, pages 293–311. Springer Verlag, 2019.

[Pet18] Tomas Petricek. What we talk about when we talk about monads. *The Art, Science, and Engineering of Programming*, 2(3):12, 2018. `doi:10.22152/programming-journal.org/2018/2/12`.

[Yor09] Brent Yorgey. Abstraction, intuition, and the “monad tutorial fallacy”, January 2009. URL: `https://archive.ph/8JmUg`.