Proof Theory / Type Theory / Category Theory Interview

https://www.knowist.ac

The following is a collection of sample questions demonstrating the level of knowledge concerning proof theory, type theory and category theory we will be expecting during our interviews.

- o -

PART 1: Proof Theory

What is proof theory?

Proof theory is the mathematics of proofs. It treats proofs as mathematical objects themselves and then explores them in a mathematical setting, trying to precisely determine characteristics of proofs (e.g. can they get "stuck" during evaluation). Proof theory plays an important role in the foundations of mathematics.

What are the kinds of structural proof theory, and which would you recommend?

There are three main ones:

- Natural Deduction
- Sequent Calculus
- Hilbert

The first two, both created by Gentzen, are the most useful in a modern world.

Name some interesting projects that use natural deduction in a practical domain that interests you.

The main paper explaining WebAssembly makes extensive use of natural deduction:

- http://delivery.acm.org/10.1145/3070000/3062363/pldi17-main166.pdf

The W3C is working on a new 3D graphics standard for the web, called WebGPU. It will come with a new shader language, called WHLSL:

- https://github.com/gpuweb/WHLSL

This is defined using OTT - a language specification system based on natural deduction:

- https://www.cl.cam.ac.uk/~pes20/ott/

Name a couple of books you have read that use natural deduction.

Harper: "Practical Foundations for Programming Languages"
https://www.cs.cmu.edu/~rwh/pfpl/2nded.pdf

Pierce: "Types and Programming Languages"
https://www.cis.upenn.edu/~bcpierce/tapl/

Give a quick explanation of natural deduction.

Natural deduction is a way of organizing judgments (pieces of knowledge), often in a hierarchical manner, using inference rules. These rules are often represented like so:

```
X      Y
_______
   Z
```

which states that if judgment X and judgement Y hold, then judgment Z holds. If a judgment always hold, then it can be presented like so:

```
_______
   Z
```

Inference rules with judgments above and below the llne are known as proper rules; those with a judgment only below the line are known as axioms.

Introduction and elimination rules can be defined to add and remove judgments from consideration.

Identify one difference between natural deduction and sequent calculus.

With natural deduction, only a single judgment appears in the conclusion below the line; with sequent calculus, multiple can.

Give a quick explanation of the relationship between natural deduction and lambda calculus.

It has been discovered that much of the logical apparatus for natural deduction in mathematics very very closely mimics lambda calculus which describes programming. They are both just different ways of loooking at the same thing. We like lambda calculus because it lets our programs run on a computer. We like natural deduction because we can use it to prove properties of our logic. Actually, we like both. Using both together gives us verified programming. This is a nice paper explaining the correspondence:

- Propositions as types

What is a proof assistant and give some examples?

A proof assistant is a tool to help create and verify mathematical proofs.
Coq: https://coq.inria.fr/
Agda: https://wiki.portal.chalmers.se/agda

What are a few books you have read on using a proof assistant?

- Software Foundations (Pierce,..): https://softwarefoundations.cis.upenn.edu/
- Certified Programming with Dependent Types (Chlipala): http://adam.chlipala.net/cpdt/

What is extraction?

When using a proof assistant, the proof itself carries more details than is strictly needed for execution. Hence many tools offer an extraction feature, which extracts out executable code from a verified proof. The idea is to first write a proof and ensure it is correct, and then have the proof assistant auto-generate code for you. Details here:

- https://coq.inria.fr/refman/addendum/extraction.html

PART 2: Type Theory

What is type theory?

Type theory is the mathematics of types. It aims to bring a coherent representation to all of mathematics using the idea of types and their values. Beginning with a very small core of ideas, type theory expands in a verifiable way to include exploring ideas such as universes, identity, equality, canonical values, normal forms, contexts, formation rules, etc. A detailed description is here:

- https://ncatlab.org/nlab/show/type+theory

Name an example of a type theory and tell me a little about it.

There are a number of type theories, the most famous of which is intuitionistic type theory, also known as Martin-Löf's Type Theory (MLTT), named after its Swedish creator. One of his famous paper on the topic is:

- "Constructive mathematics and computer programming" http://archive-pml.github.io/martin-lof/pdfs/Constructive-mathematics-and-computer-programming-1982.pdf

In type theory, what is a product and how is it denoted?

A product is a composite type. If we have A Type and B Type, then a product of both is denoted as A x B. Think of a product as AND in logic.

What is a sum and how is it denoted?

A sum is alternatives. If we have A Type and B Type, then a sum of both is denoted as A + B. Think of a sum as OR in logic.

In type theory, what is a universe?

We need a universe type to state things about all types that exist. A universe is a type that contains all the other types, except itself. If we wish to states things about that

universe and all the other types, we can create an "outer" universe containing both. This can be repeated, so we can be very flexible in structuring our type hierarchies.

What is normalization?

Expressions consisting of multiple terms can be built up in a type theory. Normalization is the idea of evaluating (or simplifying, or "reducing") more complex expressions to simple ones just consisting of simply values, known as normal forms, which cannot be simplified further.

What is Dependent Type Theory?

In traditional object-oriented programming languages (C++, C#, Java) that support generics, a type can depend on another type, such as a set of integers (`Set<int>`), or a set of Car objects (`Set<Car>`). The important concept here is what is being depended on is itself a type.

With a dependent type theory, types can depend on values. So we could have a set of 4 elements and this is defined as a type (so the 4 is part of the type definition, not just a parameter to its constructor). The practical benefit of this is to allow representation of for all (∀) and there exists (∃) from logic.

What is a Higher Inductive Type (HIT)?

A higher inductive type adds custom equality to a type, thus leading to multiple kinds of equality between pairs of elements of a type. So you may have one kind of equality we could call exact equality (sometimes represented as ≐), and these other kinds that have other characteristics (equality under custom circumstances).

What is the Univalence Axiom (UA)?

The univalence axiom identifies the strongest identity for a universe. The key goal here is that statements we prove about a type member with the custom equality also apply to the other equivalent type, thus greatly helping with mathematical reasoning.

The two ideas of HIT and UA combined are called homotopy type theory, because the initial way to describe and reason about them was using homotopy. However, this uses an axiom approach, which is not good for computation, so alternatives were sought after, leading to cubical type theory ...

Briefly, what is Cubical Type Theory?

It adds HIT and UA to type theory in a computationally friendly manner (avoiding the problematic axiom used by homotopy type theory).

Explain ...
We build a 2-dimensional cube (and later, higher dimensional structures, called n-cubes). Using equalities as our building blocks. When we get three sides of the cube, we can deduce the fourth. Similarly for higher-dimensional arrangments.

PART 3: Category Theory

What is category theory?

In mathematics, a category is an abstract mathematical universe. Category theory examines how these universes can be used to precisely describe all of mathematics.

A category is itself a mathematical object. A category consists of one or more category objects and maps between them. Sometimes terms such as morphism or arrow are used instead of map, but regardless, they represent the same concept.

Tell me more about category objects?

A category object is an assortment of elements. To begin with, it is simplest to think of a category object as a mathematical set, but in a more advanced setting other approaches are possible. The important point is that that may be 0, 1 or many elements inside a category object. So a category object is a container. When describing a mathematical universe, we need to think through about each of the elements and the category object itself (similar to members of a set and the set itself).

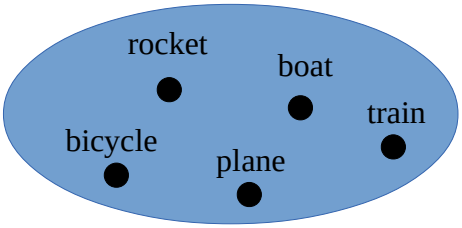
How do you represent category objects?

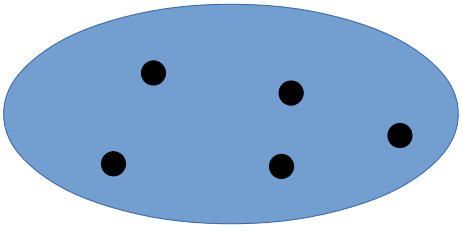
Categories are abstract so have no inherent representation. However, humans like to be able to see them and represent them, so a number of representation formats have evolved. One can be selected, depending on the space available, the level of detail required and the breadth of content to be displayed in a given space. For any category, all the representations are compatible, because they are just alternative views of the same abstract mathematical universe.

A category object can be represented in text as a collection of its elements:

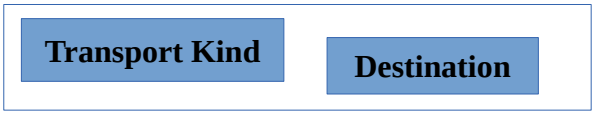
```
[boat, plane, train, bicycle, rocket]
```

Categories can be represented graphically, as internal diagrams (showing elements of its category objects, in no particular order – remember, they are set-like, not ordered sequences), with or without element names:





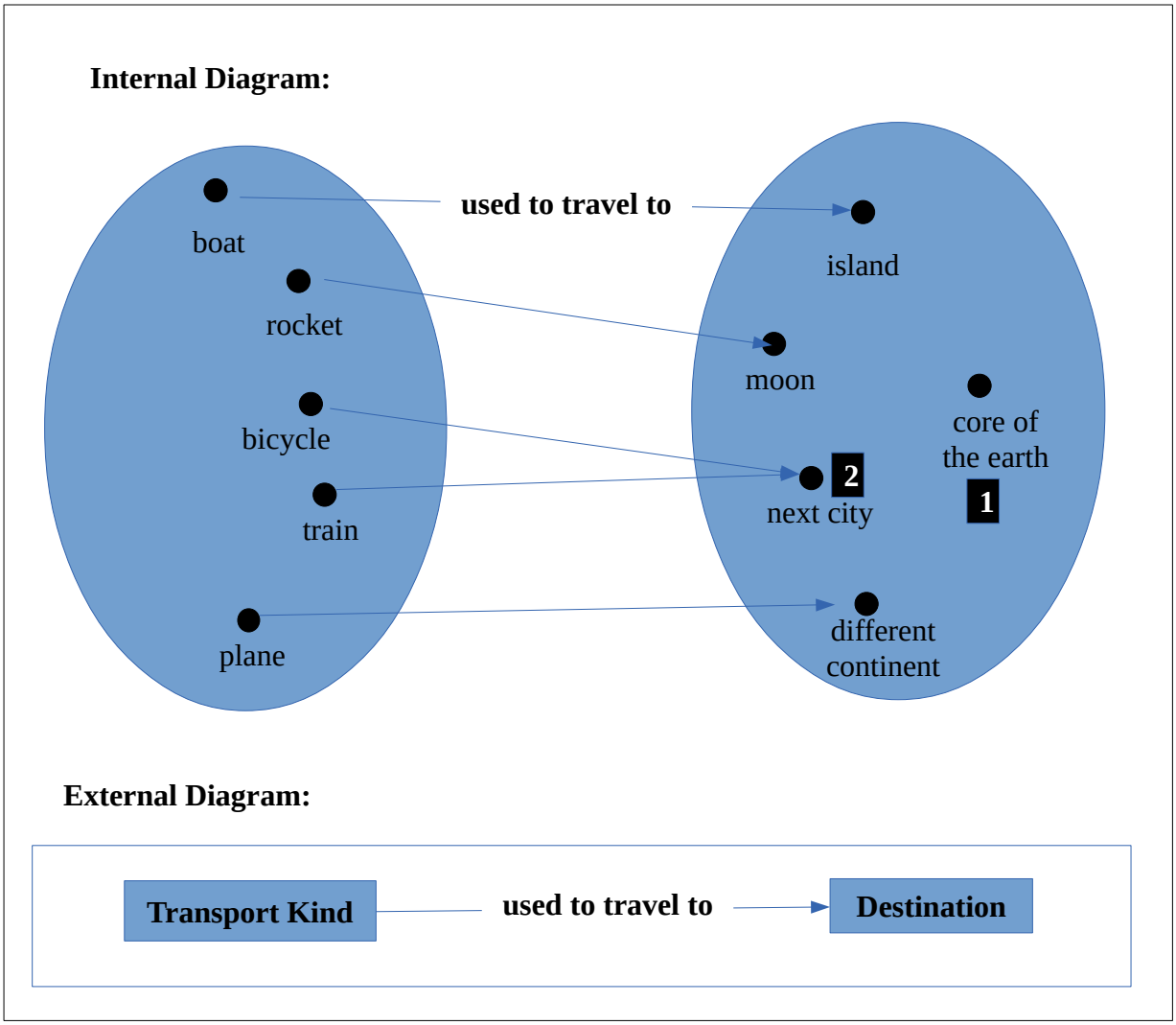
or as external diagrams (just showing the category objects themselves):



In category theory, what is a map?

A map is a directed relationship between two elements of category objects. The category object that supplies the element in the "from" part of the relationship is known as the domain; the category object that supplies the element in the "to" part of the relationship is known as the codomain.

A map is idempotent and is like a mathematical function (a pure function in programming terms), that takes one exactly input (from the domain) and returns exactly one output (from the codomain). An important rule in category theory is that, for a map from a domain to a codomain to be valid, there must exist a map from each element within the domain to any element in the codomain (the concept programmers call `NULL` does not exist). The reverse is not the case: every element of the codomain need not be a target for a map from a domain **1**. Finally, it is permitted that an individual element in the codomain is the same target for multiple maps from elements in the domain **2**, but the reverse does not hold: for each element in the domain, it must map to one and one only element in the domain.

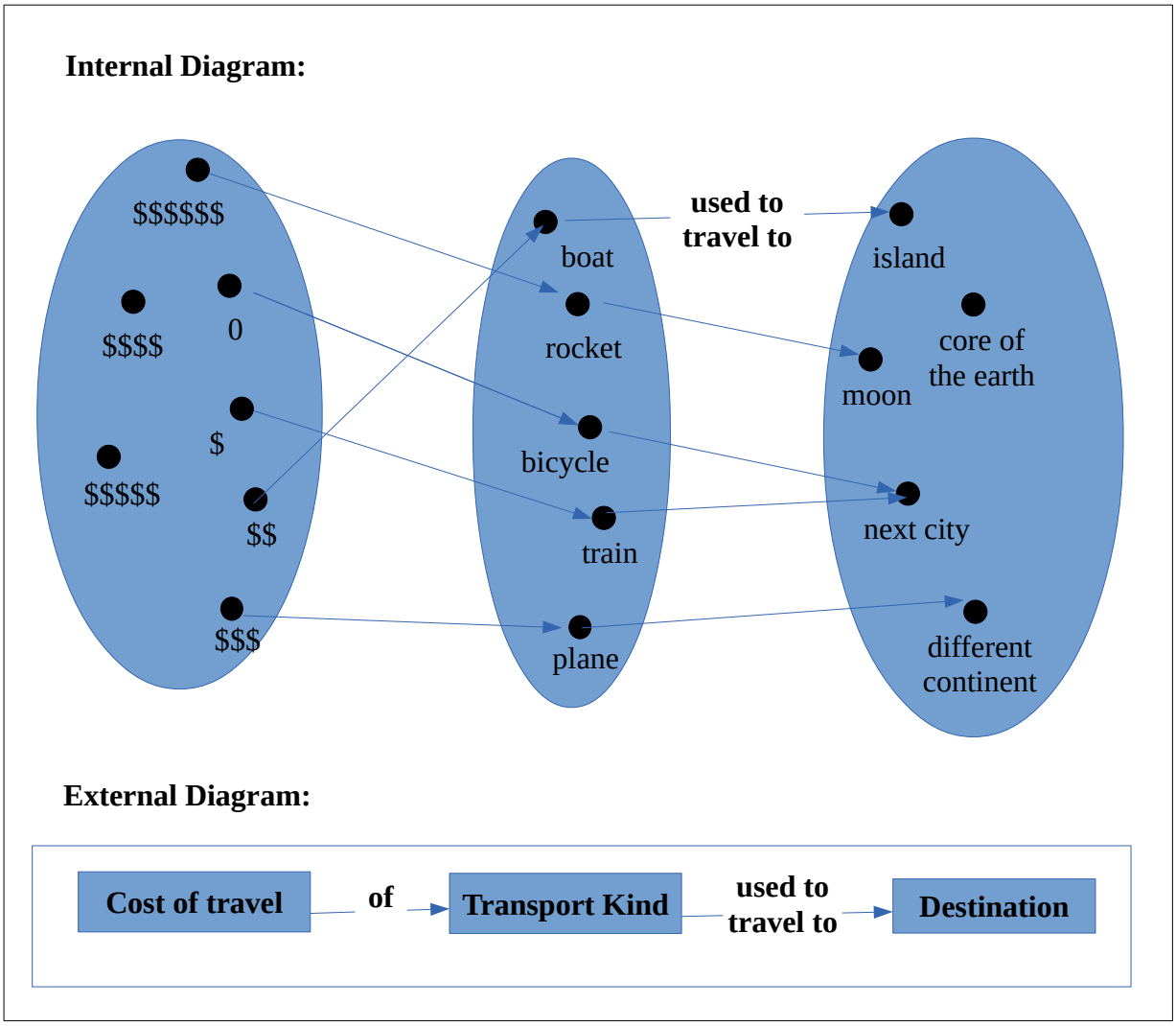


When examining larger categories, the benefits of zooming in and out using these representations are obvious.

Tell me about map composition.

Maps can be composed, thus creating a path.

Where there are multiple category objects in a category and maps between multiple pairs of category objects, then it is possible to compose those maps. This is best demonstrated with a sample:



This category tells you getting to the moon is expensive. Its does this by telling you the use of a rocket costs lots ($$$$$$$) and then saying to get to the moon you need to use a rocket. To represent composition in text, the letter o is used. So if we use x and y to represent the above two mappings, then: `y ∘ x` represents this path (read as `y follows x`). If we imagine both `x` and `y` are functions, it is like function composition, as `y ∘ x` means `y(x())` which means execute `x` first, then `y`.

What is an endomap?

The term ‘endo’ is Greek for into itself or internal, as see in terms such as endoscope, which a doctor uses to examine a patient. In category theory, an endomap or endomorphism is a mapping where the domain and codomain of the map are the same category object.

What is an identity map?

An identity map is an endomap, whereby in addition to the domain and codomain being the same category object, the ‘from’ element is the same as the ‘to’ element in each map instance (so a mapping from an element to itself).

What is a functor and given an example?

Categories can be arranged hierarchically (one category contained with another category), in which case the contained categories appear as category objects in the parent category. There can be structure preserving mappings between such contained categories, and these mappings are called functors.

Presheaves are examples of functors.

What are natural transformations?

Functors can also play the role of category objects and then the strructural preseving mappings between them are known as natural transformations.

What is a section and a retraction?

A retraction for a map is another map for which when both are composed gives us the identy mapping of the codomain. A section is the opposite.

What are initial objects and terminal objects?

Terminal objects are those for which, for every category object in the catgory, there a single map from it to the terminal object. Initial objects are the reverse.