

What is a proof?

Nathanael Arkor

(DIMEA/FORMELA 04.04.22)

# What is a proof? (And why should we care?)

There are two traditional meanings of the term 'proof' in mathematics.

**Informal:** an argument that suffices to convince other mathematicians of the truth of some statement.

**Formal:** a derivation of some statement from axioms and rules in accordance with a formal system.

Conceptually, an argument is convincing if we believe we could, in theory, construct a corresponding formal proof.



A formal proof is entirely rigorous, while an informal proof often has some reasoning left implicit or ambiguous ('abuses of notation', etc.).

## A concerning problem

In practice, this conventional, informal manner of proof works quite well. However, it is **not perfect**.

Often small mistakes slip by reviewers into published work.

Sometimes much larger problems...

# A harrowing tale

Voevodsky

Kapranov and I decided that we know what the definition should be and how to prove the conjecture with this definition.

$\infty$ -groupoids  
model homotopy  
types

$\infty$ -groupoid

We wrote a paper with a sketch of the proof and published it in one of the best Russian mathematical journals and the paper with the complete proof was published in the proceedings of the conference that I have been invited to.

Then in 2003, twelve years after our proof was published in English, a preprint appeared on the web in which his author, Carlos Simpson, very politely claimed that he has constructed a counter-example to our theorem.

I was busy with the work on the motivic program and very sure that our proof is correct and ignored the preprint.

And then in the Fall of 2013, less than a year ago, some sort of a block in my mind collapsed and I suddenly understood that Carlos Simpson was correct and that the proof which Kapranov and I published in 1991 is wrong.

Not only the proof was wrong but the main theorem of that paper was false!

(From 'How I became interested in foundations of mathematics', Voevodsky, 2014.)

## A (partial) solution?

While some issues arise due to the informal nature of our proofs, the larger issue is human fallibility.

Put simply: humans make mistakes.

## A (partial) solution?

While some issues arise due to the informal nature of our proofs, the larger issue is human fallibility.

Put simply: **humans make mistakes.**

How can we mitigate this issue?



## A (partial) solution?

While some issues arise due to the informal nature of our proofs, the larger issue is human fallibility.

Put simply: **humans make mistakes.**

How can we mitigate this issue? As computer scientists, there is an obvious potential solution...

## Computer-verified proofs

What if we could leave writing proofs for humans, but have the proofs verified by a computer?

## Computer-verified proofs

What if we could leave writing proofs for humans, but have the proofs verified by a computer?

Not only would this eliminate erroneous proofs, it would also speed up the frustratingly slow review process, as human reviewers would only have to evaluate the significance and exposition of a paper, not its correctness.

In the ideal scenario, a computer could check a **natural language proof**. However, this is a hard problem, so it is reasonable to start with a smaller first step: **verification of formal proofs**.

In the ideal scenario, a computer could check a **natural language proof**. However, this is a hard problem, so it is reasonable to start with a smaller first step: **verification of formal proofs**.

If we could solve this problem, then 'all' that would remain is to mechanise the informal  $\rightsquigarrow$  formal step.

## The question of foundations

For computers therefore to validate mathematical proofs, we need to decide upon a formal system of proof.

## The question of foundations

For computers therefore to validate mathematical proofs, we need to decide upon a formal system of proof.

Which to choose?

## The question of foundations

For computers therefore to validate mathematical proofs, we need to decide upon a formal system of proof.

Which to choose?

Traditionally, modern mathematics has been based upon ZFC. Type theorists may instead advocate a variant of Martin-Löf Type Theory.

There are many options...



## Synthetic mathematics

There is an alternative school of thought that proposes, instead of **encoding** everything in a single framework, which can lead to complex and obfuscated definitions, we ought to directly **axiomatise** the structures we are interested in.

## Synthetic mathematics

There is an alternative school of thought that proposes, instead of **encoding** everything in a single framework, which can lead to complex and obfuscated definitions, we ought to directly **axiomatise** the structures we are interested in.

For instance, in **synthetic geometry** one axiomatises points, lines, curves, etc. and their relationships, rather than encoding these as subsets of points in a plane.

Synthetic mathematics often suggests (or even forces) more elegant or conceptual proofs than traditional analytic mathematics.

Synthetic mathematics often suggests (or even forces) more elegant or conceptual proofs than traditional **analytic mathematics**.

What if we were to formulate a system of proof taking synthetic mathematics seriously?

In other words, rather than imposing a specific system of proof, provide the flexibility to work within different systems according to need.

To understand this proposal, let us analyse various systems of proof, and identify their commonalities.

To understand this proposal, let us analyse various systems of proof, and identify their commonalities.

In essence, we shall see that all can be understood to be built up from deductions of the form

premisses  $\Rightarrow$  conclusion

for different meanings of premiss and conclusion.

# Equational logic / universal algebra

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash f(t_1, \dots, t_n)} \quad (\text{Term formation})$$

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash s = s'} \quad (\text{Equality})$$

(+ Rules governing behaviour of equality)

# Equational logic / universal algebra

premisses

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\text{ (Term formation)}}$$

conclusion  $\vdash f(t_1, \dots, t_n)$

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash s = s'} \quad \text{(Equality)}$$

(+ Rules governing behaviour of equality)



# Equational logic / universal algebra

premisses

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash f(t_1, \dots, t_n)}$$

(Term formation)

conclusion

$$\vdash f(t_1, \dots, t_n)$$

1. "t is a term"

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash s = s'}$$

(Equality)

(+ Rules governing behaviour of equality)

# Equational logic / universal algebra

premisses

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash f(t_1, \dots, t_n)}$$

(Term formation)

conclusion  $\vdash f(t_1, \dots, t_n)$

$$\frac{\vdash t_1 \quad \dots \quad \vdash t_n}{\vdash s = s'}$$

$\vdash s = s'$

(Equality)

1. "t is a term"
2. "the terms t and t' are equal"

(+ Rules governing behaviour of equality)

# Type theory

$$\frac{\cdot \text{ ctx}}{\Gamma \text{ ctx} \quad A \text{ type}} \\ \hline \Gamma, a:A \text{ ctx}$$

$$\frac{\Gamma \vdash t_1:A_1 \quad \dots \quad \Gamma \vdash t_n:A_n}{\Gamma \vdash f(t_1, \dots, t_n):B}$$

$$\frac{A \text{ type}}{\cdot, a:A \vdash a:A}$$

(+ Rules for equality, substitution, weakening, etc.)

# Type theory

$$\frac{\cdot \text{ ctx}}{\Gamma \text{ ctx} \quad A \text{ type}} \\ \hline \Gamma, a:A \text{ ctx}$$

$$\frac{\begin{array}{l} \Gamma \text{ ctx} \quad B \text{ type} \\ A_1 \text{ type} \quad \dots \quad A_n \text{ type} \\ \Gamma \vdash t_1:A_1 \quad \dots \quad \Gamma \vdash t_n:A_n \end{array}}{\Gamma \vdash f(t_1, \dots, t_n): B}$$

$$\frac{A \text{ type}}{\cdot, a:A \vdash a:A}$$

(+ Rules for equality,  
substitution, weakening,  
etc.)

# Type theory

$\cdot$  ctx  
 $\Gamma$  ctx    A type  
 $\Gamma, a:A$  ctx

$\Gamma$  ctx    B type  
A<sub>1</sub> type ... A<sub>n</sub> type  
 $\Gamma \vdash t_1:A_1 \dots \Gamma \vdash t_n:A_n$   
 $\Gamma \vdash f(t_1, \dots, t_n):B$

A type  
 $\cdot, a:A \vdash a:A$

(+ Rules for equality, substitution, weakening, etc.)  
**1. Contexts**

# Type theory

$$\frac{\cdot \text{ ctx}}{\Gamma \text{ ctx} \quad \text{A type}} \quad \Gamma, a:A \text{ ctx}$$

$$\frac{\Gamma \text{ ctx} \quad \text{A}_1 \text{ type} \quad \dots \quad \text{A}_n \text{ type} \quad \Gamma \vdash t_1:A_1 \quad \dots \quad \Gamma \vdash t_n:A_n}{\Gamma \vdash f(t_1, \dots, t_n): B} \quad \text{B type}$$

$$\frac{\text{A type}}{\cdot, a:A \vdash a:A}$$

(+ Rules for equality, substitution, weakening, etc.)

1. Contexts
2. Types

# Type theory

$$\frac{\cdot \text{ ctx}}{\Gamma \text{ ctx} \quad A \text{ type}} \\ \hline \Gamma, a:A \text{ ctx}$$

$$\begin{array}{l} \Gamma \text{ ctx} \quad B \text{ type} \\ A_1 \text{ type} \quad \dots \quad A_n \text{ type} \\ \hline \Gamma \vdash t_1:A_1 \quad \dots \quad \Gamma \vdash t_n:A_n \\ \hline \Gamma \vdash f(t_1, \dots, t_n):B \end{array}$$

$$\frac{A \text{ type}}{\cdot, a:A \vdash a:A}$$

(+ Rules for equality, substitution, weakening, etc.)

1. Contexts
2. Types
3. Terms

# Type theory

$$\frac{\cdot \text{ ctx}}{\Gamma \text{ ctx} \quad A \text{ type}} \\ \hline \Gamma, a:A \text{ ctx}$$

$$\frac{\begin{array}{l} \Gamma \text{ ctx} \quad B \text{ type} \\ A_1 \text{ type} \quad \dots \quad A_n \text{ type} \\ \Gamma \vdash t_1:A_1 \quad \dots \quad \Gamma \vdash t_n:A_n \end{array}}{\Gamma \vdash f(t_1, \dots, t_n): B}$$

$$\frac{A \text{ type}}{\cdot, a:A \vdash a:A}$$

(+ Rules for equality, substitution, weakening, etc.)

1. Contexts
2. Types
3. Terms
4. Equality



## Judgement structure

We said that deductions can be seen as having the form

premisses  $\Rightarrow$  conclusion

but where premisses and conclusions can mean different things depending on the logic in question.

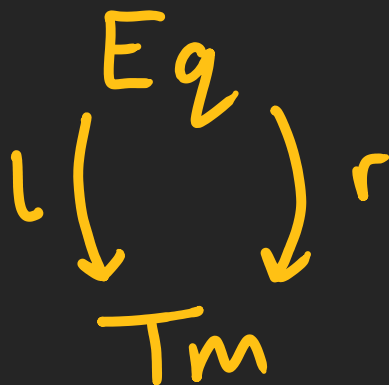
We shall call the 'shape' of a premiss or conclusion a judgement.

For instance, in **equational logic** there are two judgements:

1. **—** is a term.

2. The terms **—** and **—** are equal.

We may represent this structure with a graph

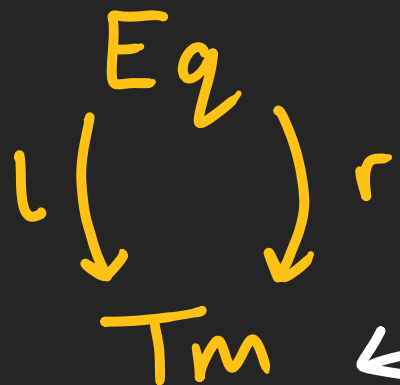


For instance, in **equational logic** there are two judgements:

1. **—** is a term.

2. The terms **—** and **—** are equal.

We may represent this structure with a graph



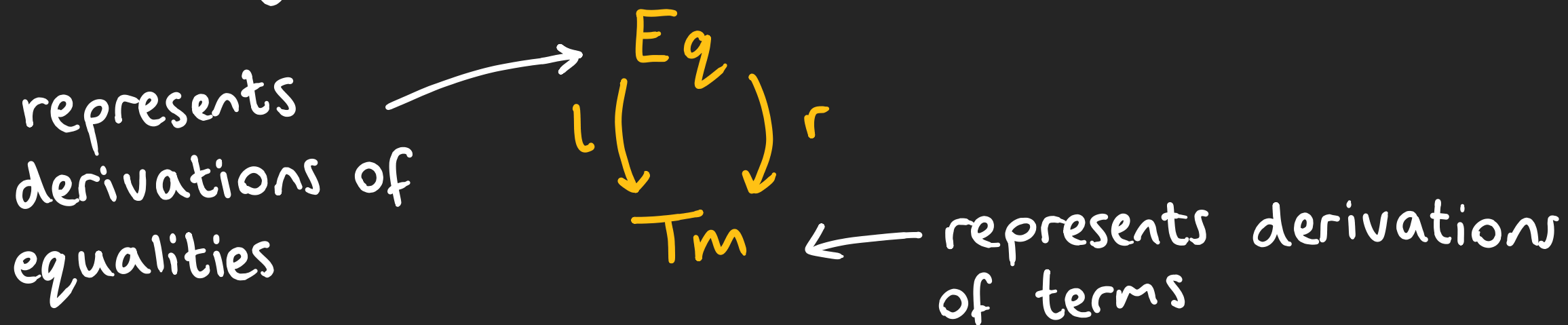
← represents derivations of terms

For instance, in **equational logic** there are two judgements:

1. **—** is a term.

2. The terms **—** and **—** are equal.

We may represent this structure with a graph

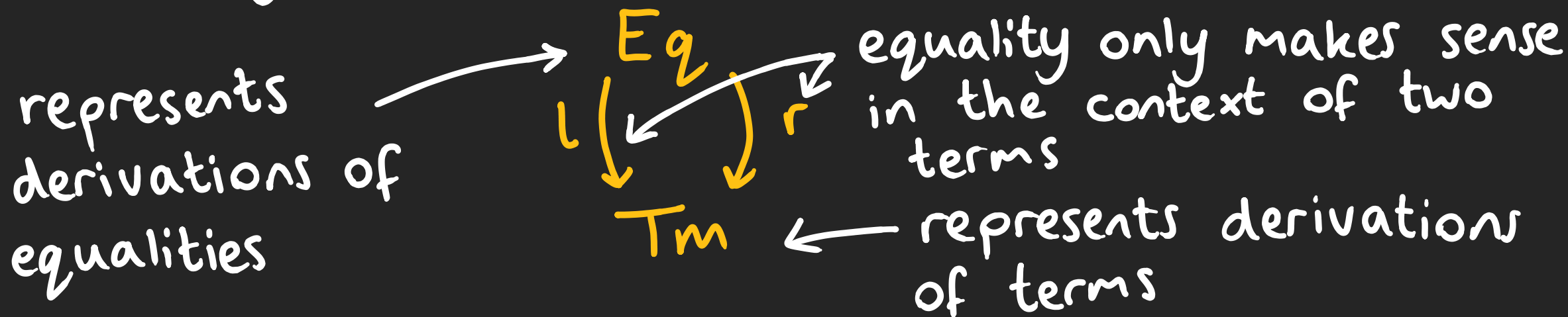


For instance, in **equational logic** there are two judgements:

1. **—** is a term.

2. The terms **—** and **—** are equal.

We may represent this structure with a graph



In a (simple) type theory, there are four judgements:

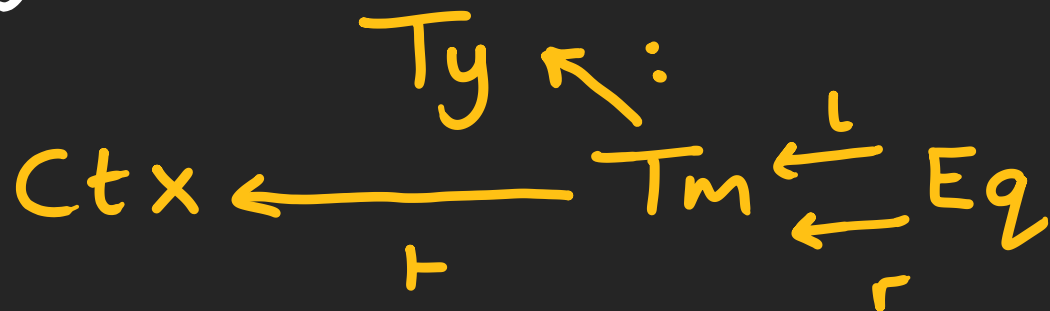
1.  $\_$  is a context.

2.  $\_$  is a type.

3.  $\_$  is a term of type  $\_$  in context  $\_$ .

4. The terms\*  $\_$  and  $\_$  are equal.

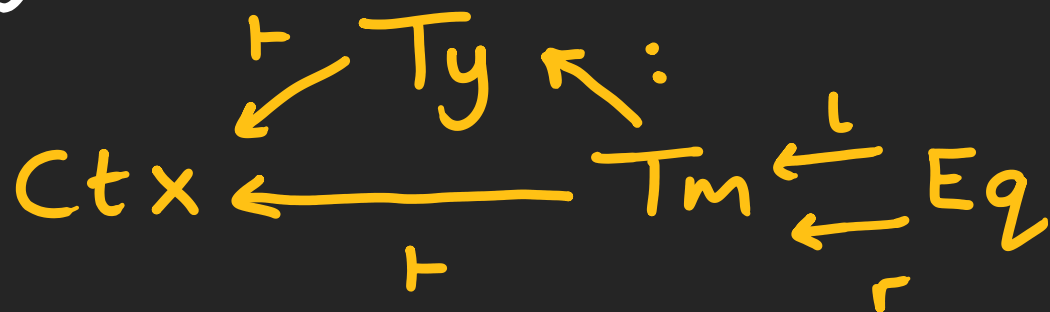
We may represent this structure with a graph



In a (dependent) type theory, there are four judgements:

1.  $\_$  is a context.
2.  $\_$  is a type in context  $\_$ .
3.  $\_$  is a term of type  $\_$  in context  $\_$ .
4. The terms\*  $\_$  and  $\_$  are equal.

We may represent this structure with a graph



(plus a path equality condition)

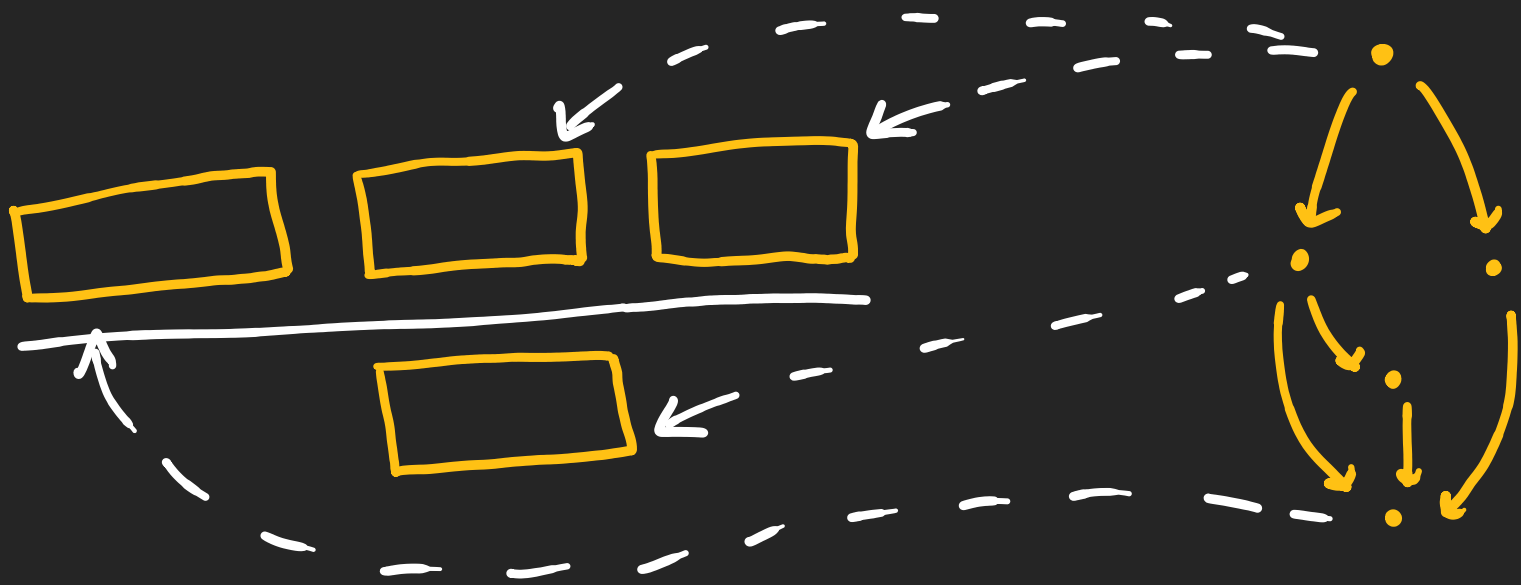
A **judgement structure** is an acyclic category with finite fan-out.

This means we have an acyclic directed multigraph in which we may declare certain paths to coincide, closed under composition, and such that





Each **vertex** of a judgement structure defines a possible shape for a premiss or conclusion.



Each **edge** defines the dependencies for the respective premiss or conclusion.

## Intuition for judgements

We can think of judgements as the different structures a formal system of proof can talk about.

## Intuition for judgements

We can think of **judgements** as the different **structures** a formal system of proof can talk about.

On the other hand, the **rules** describe how we can **form** such structures. In general, a rule can assume the existence of any of the structures to form any other structure.

## An aside on multiple conclusions

I am considering deductions with multiple premisses, but only a single conclusion.

This is because a deduction of the form

$$\frac{P_1 \cdots P_n}{C_1 \cdots C_m}$$

is equivalent to a sequence of deductions

$$\frac{P_1 \cdots P_n}{C_1} \cdots \frac{P_1 \cdots P_n}{C_m}$$

## Deduction systems

**Judgement structures** define the essential structure of a framework for proof: the shapes that its premisses and conclusions can take.

A **rule structure** defines the possible deductive steps that may be taken to form a proof.

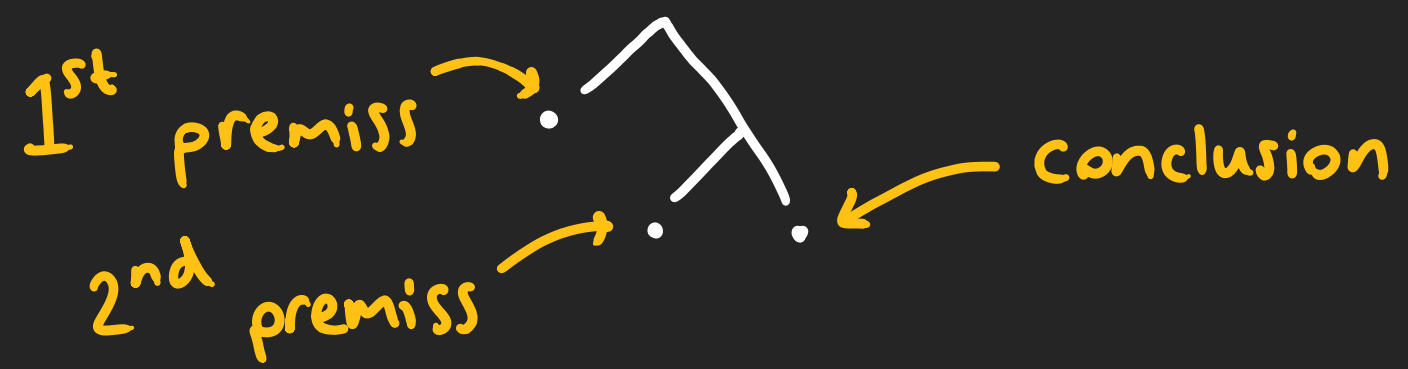
Together, these form a **deduction system**, which defines a framework for proof.

## Rule structure

To define a **deductive rule**, we must specify the structure of the premisses and conclusion.

For each, we declare a **judgement**, and specify its **dependencies**.





The theory of a monoid action is an extension of universal algebra. Hence, the judgement structure is given by

$$T_y \leftarrow T_m \leftarrow E_q$$



The theory of a monoid action is an extension of universal algebra. Hence, the judgement structure is given by

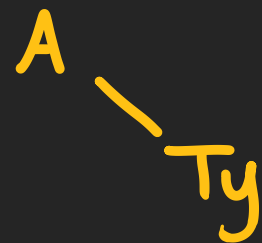
$$Ty \leftarrow Tm \leftarrow Eq$$

The rule

$$\frac{}{M \text{ type}}$$

$$\frac{}{A \text{ type}}$$

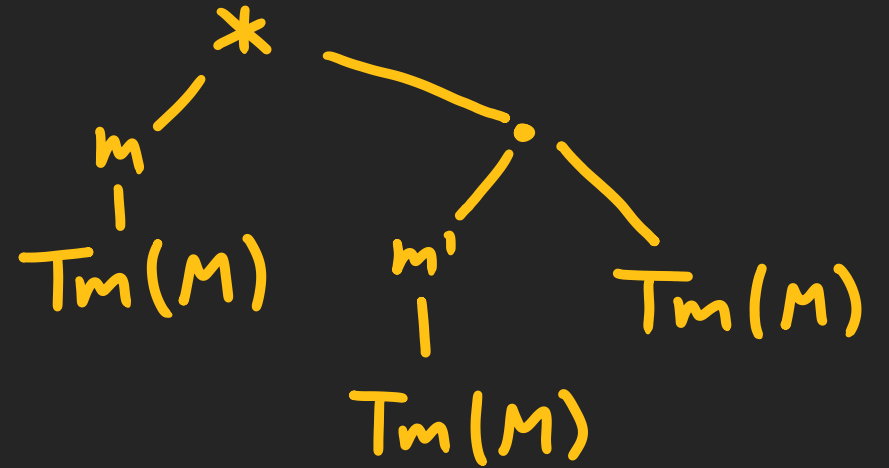
is represented by



The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

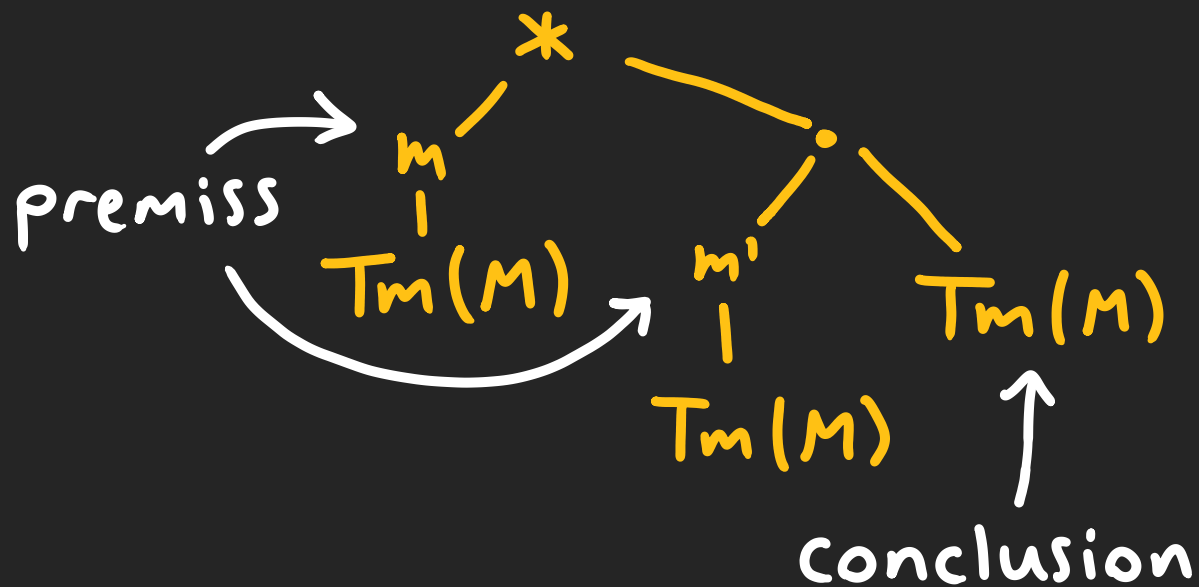
is represented by



The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

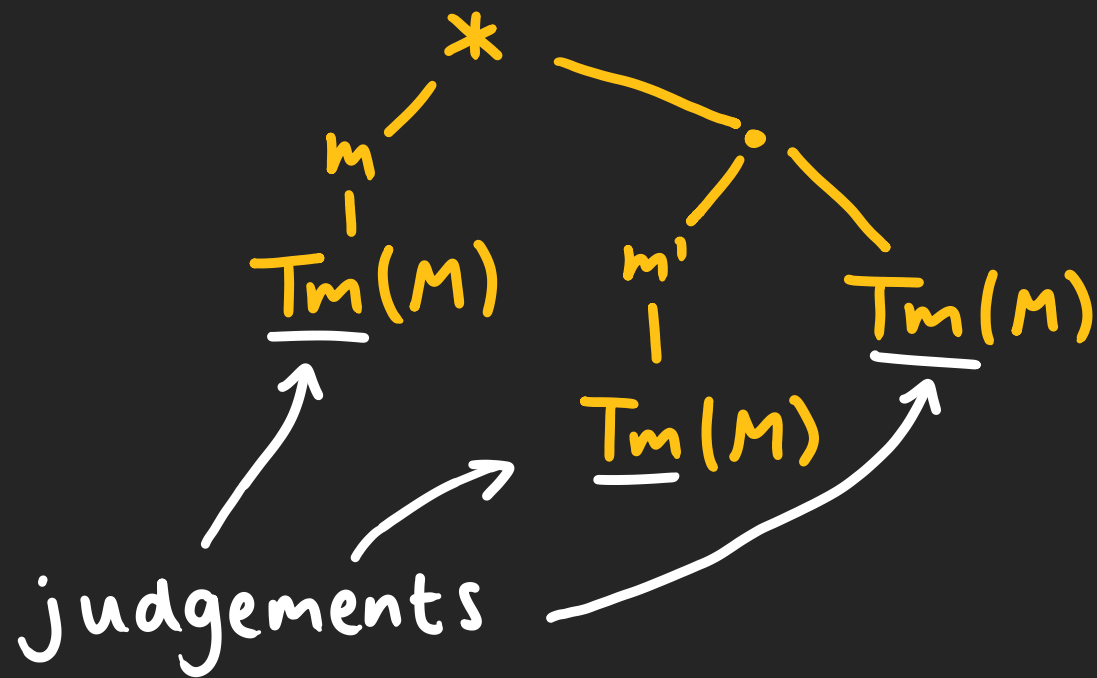
is represented by



The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

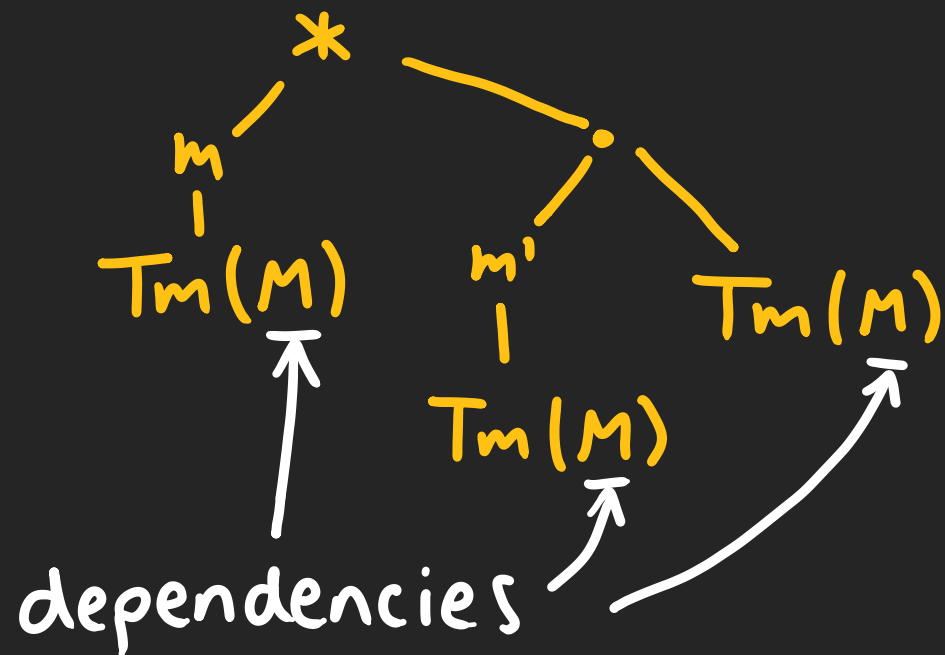
is represented by



The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

is represented by

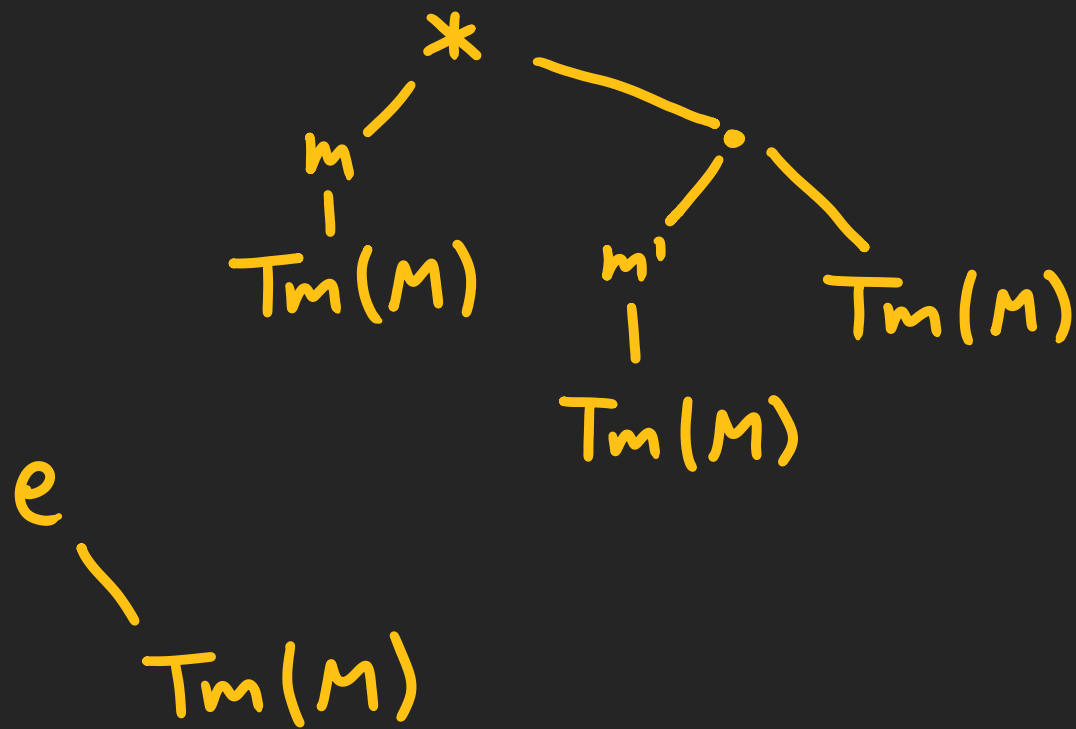


The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

$$\frac{}{\vdash e : M}$$

is represented by



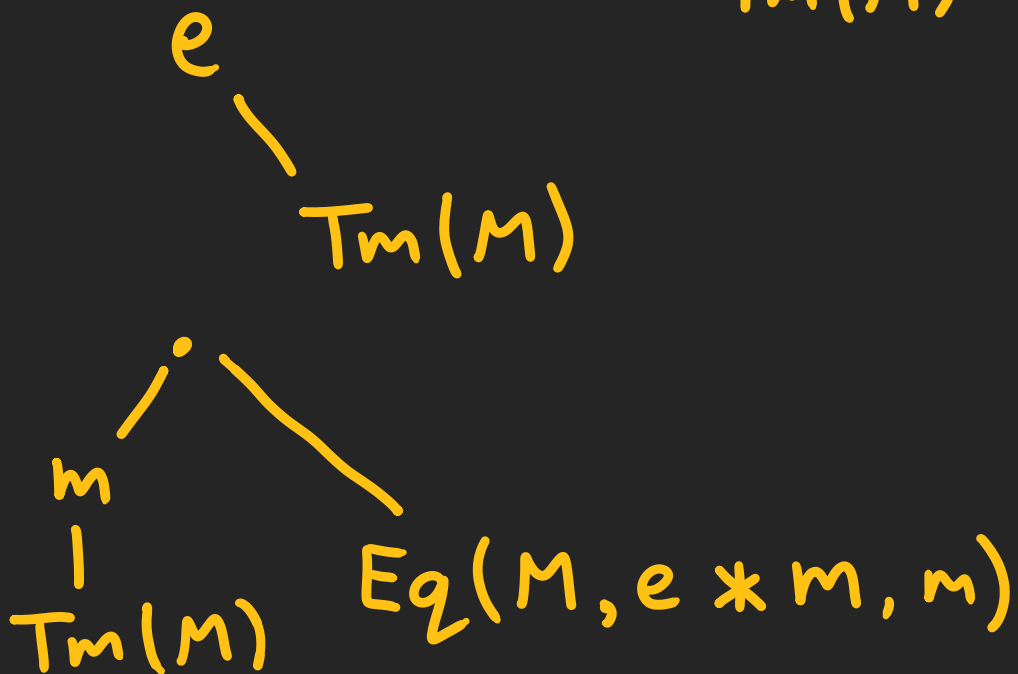
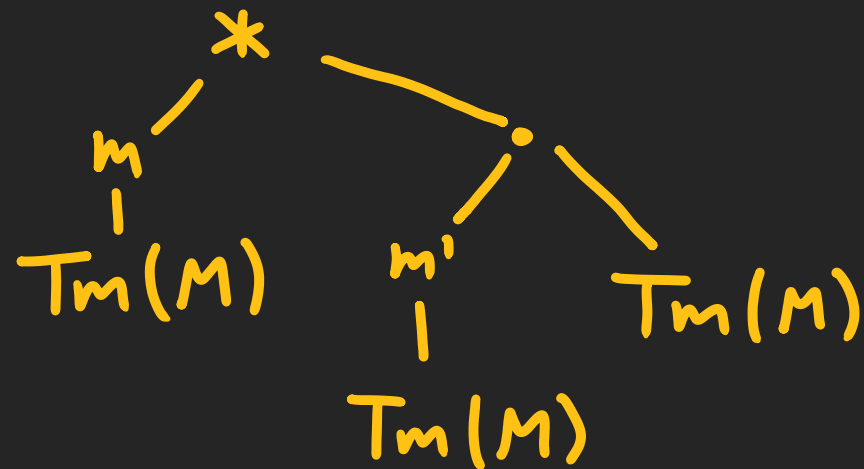
The rule

$$\frac{\vdash m : M \quad \vdash m' : M}{\vdash m * m' : M}$$

$$\frac{}{\vdash e : M}$$

$$\frac{\vdash m : M}{\vdash e * m = m}$$

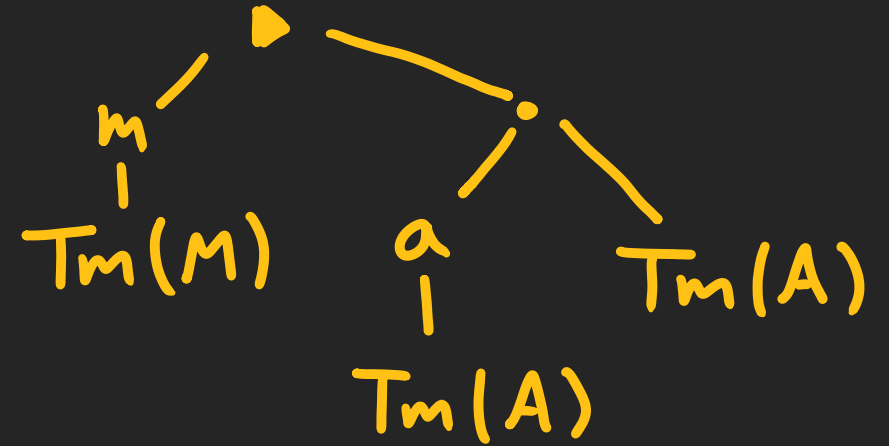
is represented by



The rule

$$\frac{\vdash m : M \quad \vdash a : A}{\vdash m \blacktriangleright a : A}$$

is represented by





## The shape of a rule

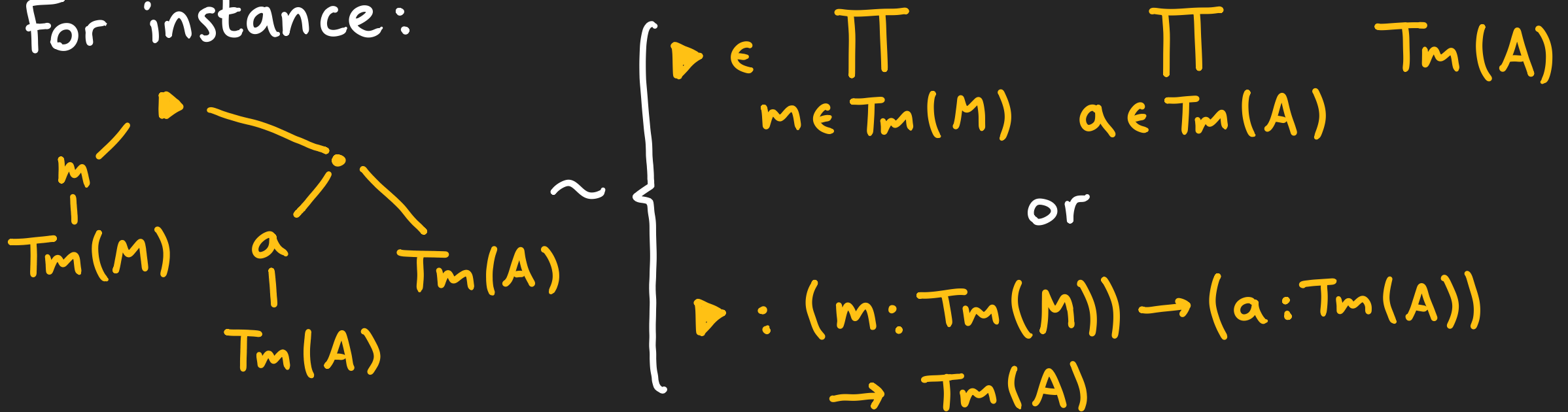
A rule structure is given by a list of rule trees. A rule tree is a binary tree with labelled leaves. (The branches are technically unlabelled, but we can think of them as being labelled by the names of the premisses.)

Each leaf is given by a judgement, and by terms specifying the dependencies of that judgement.

We can think of a **rule tree** as an **operator** on judgements: the **terms** are built inductively from the **previous rule trees**, as well as the **prior branches** of the **present rule tree**.

We can think of a rule tree as an **operator** on judgements: the terms are built inductively from the previous rule trees, as well as the prior branches of the present rule tree.

For instance:



We can think of a **rule tree** as an **operator** on judgements: the **terms** are built inductively from the **previous rule trees**, as well as the **prior branches** of the present rule tree.

For instance, in  $Eq(M, e * m, m)$ :

- $M$  is the first rule tree.
- $e * m$  is formed from  $e$ , the third rule tree;  $m$ , the first branch; and  $*$ , the second rule tree.
- $m$  is the first branch.

In other words, a term defined in a **rule tree** can refer to any of the following vertices:



In a sense to be clarified later, we can represent the **list of rule trees** as a single tree:



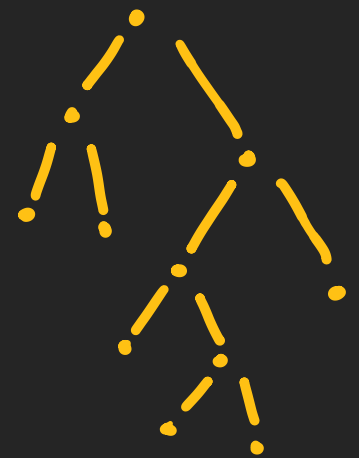
A **rule structure** is thus equivalently given by a single **rule tree**, and this is how we will think of them.

In summary, a **deduction system** is specified by two graph-like structures:



Judgement structure

Rule structure



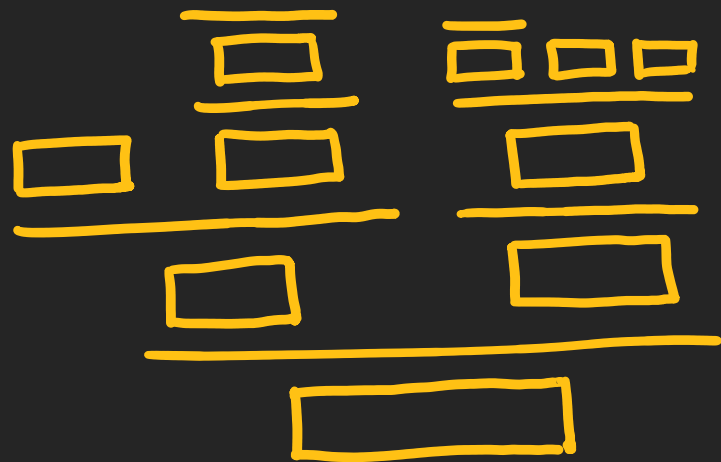
# Proofs and deduction systems

A **proof** in a formal system of proof is given by **grafting** deductive rules.



# Proofs and deduction systems

A proof in a formal system of proof is given by grafting deductive rules.



This corresponds precisely to term formation in the corresponding rule structure.

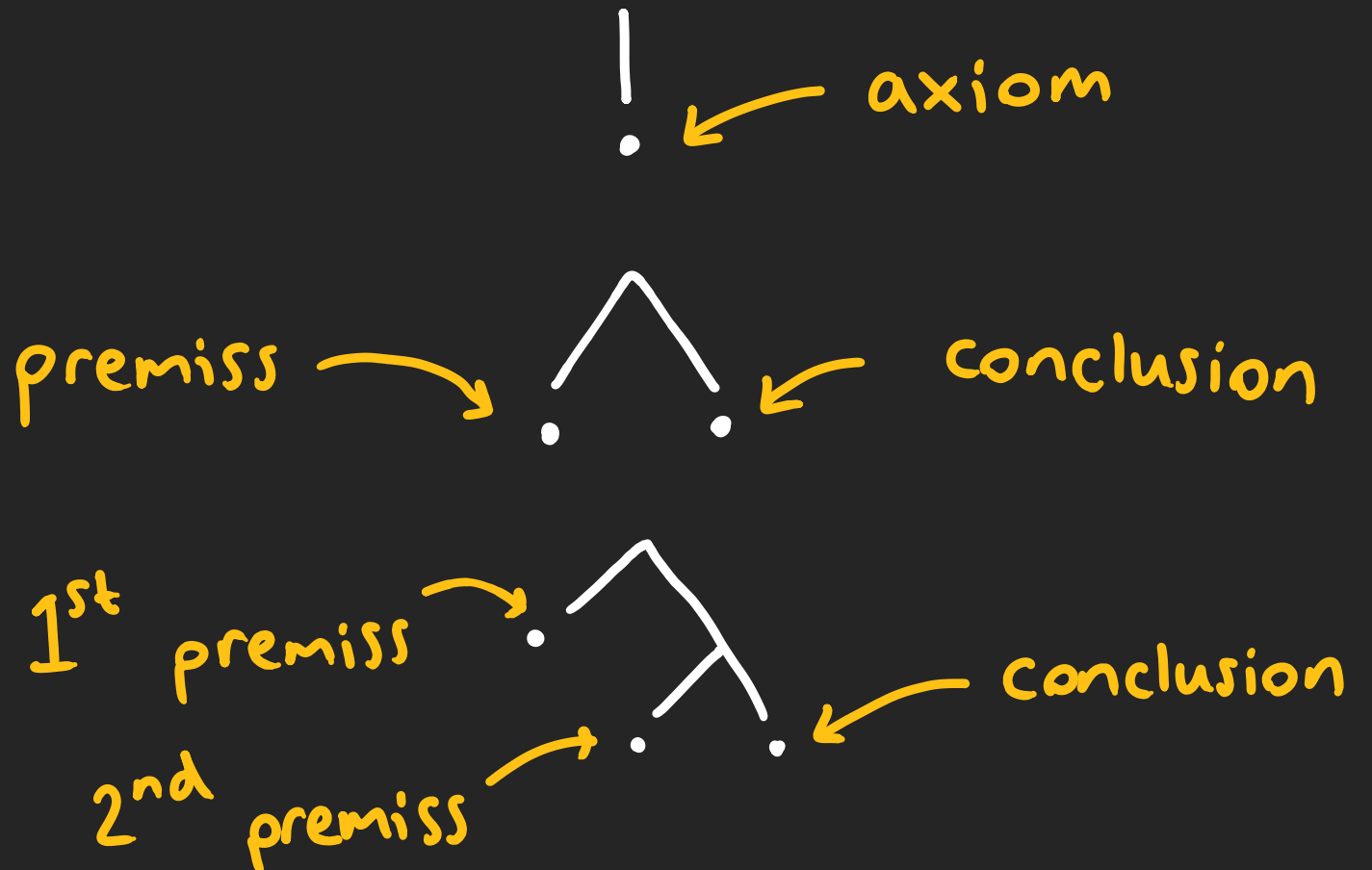


## A Curry-Howard correspondence

This idea that **proofs** correspond to **terms** is an old idea in type theory, and the structure of **deduction systems** can be seen to correspond to a certain type theory.

Recall that **rule trees** are defined to be **arbitrary binary trees**, but our examples all have a certain form...

What about other branching structures?



## Higher-order rules

An operad has, for each  $n \in \mathbb{N}$ , an  $n$ -ary operator

$$\frac{\vdash t_1 : O \quad \dots \quad \vdash t_n : O}{\vdash \otimes_n(t_1, \dots, t_n) : O}$$

Traditionally, this is viewed as an **axiom scheme**: an infinite family of operators. However, we may more elegantly view this as a **higher-order operator**.

Let us rewrite

$$\frac{\vdash t_1 : 0 \quad \dots \quad \vdash t_n : 0}{\vdash \otimes_n(t_1, \dots, t_n) : 0}$$

as

$x : \mathbb{N} \vdash t : 0$   $\xrightarrow{\text{t is now parameterised (internally) by an } n}$

$$\frac{x : \mathbb{N} \vdash t : 0}{\vdash \otimes(x.t) : 0}$$

Let us rewrite

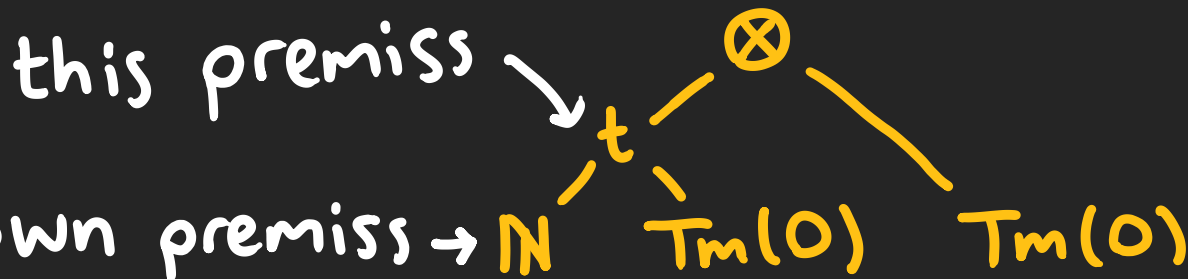
$$\frac{\vdash t_1 : 0 \quad \dots \quad \vdash t_n : 0}{\vdash \otimes_n(t_1, \dots, t_n) : 0}$$

as


$$\frac{x : \mathbb{N} \vdash t : 0}{\vdash \otimes(x.t) : 0}$$

*t is now parameterised (internally) by an n*

We can represent this as a rule tree:




A **rule tree** with complex branching corresponds to a **higher-order rule**: a rule whose premisses can be other rules.

A tree  can be thought of as an operator  $(1 \rightarrow 2 \rightarrow 3) \rightarrow 4 \rightarrow 5$ .

higher-order

A **rule tree** with complex branching corresponds to a **higher-order rule**: a rule whose premisses can be other rules.

A tree  can be thought of as an operator  $(1 \rightarrow 2 \rightarrow 3) \rightarrow 4 \rightarrow 5$ .

higher-order

However, in typical logics, it is unusual to see operators with order  $> 3$ .

## So what?

We have a definition of **deduction system** that captures many systems of proof. This is all good and well theoretically, but is it useful practically?



## So what?

We have a definition of **deduction system** that captures many systems of proof. This is all good and well theoretically, but is it useful practically?

Checking a term is **well-formed** in a deduction system is equivalent to **checking a proof** in the given system of proof. So an implementation of a **validation algorithm for deduction systems** gives us a way to mechanically verify proofs.

In fact, it is possible to implement an algorithm for validating deduction systems very concisely (~200 lines of Rust).

In fact, it is possible to implement an algorithm for validating deduction systems very concisely (~200 lines of Rust).

This makes it feasible to prove correctness of an implementation, and encourages multiple implementations — something that is not generally true for other computer proof systems.

Demo time

(if time permits)

## Relation to existing proof assistants

There exist systems for mechanically verifying proofs, which are already used by some mathematicians: Agda, Coq, Lean, Isabelle, etc.

However, the approach I have presented has several strengths over these systems.

- Existing systems tend to choose a single system of proof, giving them less flexibility, and requiring definitions and theorems to be encoded.
- Consequently, the systems are not inter-compatible: a theorem proven in one system cannot be used in another.
- These proof systems are complex, requiring large **proof kernels**, which are hard to verify.

## Next steps (1/2)

The strength in the definition of **deduction system** is its simplicity and expressivity.

However, in practice, this simplicity has a drawback: complex definitions and proofs can be tiresome to write and read, because everything is **explicit**.

We would like a **higher-level language** to manipulate deduction systems.

## Next steps (2/2)

Just as crucial are the theoretical aspects: how can we be confident that **deduction systems** are well-behaved: in other words, that they are a sensible mathematical structure for proof?

**Deduction systems** can be given a **categorical semantics** in structures related to **locally cartesian-closed categories**, which in a suitable sense justifies their study.



## Conclusion

- Computers have the potential to improve the way we do mathematics through proof verification.
- A synthetic perspective on mathematics leads naturally to a highly expressive formal deductive system, which captures many existing systems of proof.
- Deduction systems can be implemented very concisely, making them an ideal proof kernel candidate.

## Epilogue

There is much more that can be said regarding this formalism, particularly in its relationship to other ideas, e.g.

- ( $\lambda$ -free) logical frameworks
- Contextual categories
- Type theory in type theory

but this requires more background than I have time for here, so I shall simply mention them.