Interactive Proof Assistants

Thierry Coquand

University of Gothenburg

Content

Why is it possible to check mathematical correctness using computers?

Why is it can be interesting?

How does it look like?

Correctness?

How to know if something produced by chatgpt is correct?

Ouside mathematics: have to rely on an external authority

In mathematics the situation is different: there is an *objective* and purely internal way to ensure correctness of a mathematical argument

This can be done only using the *form* and not the *content* of the argument

Mathematical Correctness

The fact that *logical* reasoning could be recognised correct only by the form and not the content of an argument was a key insight of Aristotle (300 BC)

That such *formal* rules could be described in complete details for *mathematics* is a relatively recent discovery: Frege (1879), Russell-Whitehead (1910), Hilbert (1920s)

This does not mean that intuitions, social aspects don't play a crucial role in mathematics but at the end, a mathematical argument is only accepted if it is *formally* correct, through an objective process that has been slowly (and is still being) made explicit

Mathematical Correctness

Mathematicians are being more and more precise in presenting their proofs

Before the advent of computers, to obtain abolute precision in mathematics was considered to be purely ideal and to be unfeasible, e.g. by the group Bourbaki (1940-), who attempted to have a presentation of mathematics "as completely precise as possible"

The situation has changed with computers: it is now possible to write even complex proofs in complete details using *interactive proof assistants*, software that help mathematicians/computer scientists in writing mathematical arguments

Interactive Proof Assistants: Why?

Why one may want to do this?

Some proofs required in computer science (software correctness) are too long and complex to be sure of all details if carried out using pencil and paper

For some applications, to have all details correct is essential

Interactive Proof Assistants: Computer Science

Two examples (more in Greg Morrissett's talk)

seL4: a high-assurance, high-performance operating system microkernel *seL4's implementation is formally (mathematically) proven correct (bug-free) against its specification, has been proved to enforce strong security properties, and if configured correctly its operations have proven safe upper bounds on their worst-case execution times*

CompCert: formal verification of realistic compilers usable for critical embedded software. *Such verified compilers come with a mathematical, machine-checked proof that the generated executable code behaves exactly as prescribed by the semantics of the source program.*

Interactive Proof Assistants: Mathematics

In mathematics, some recent proofs are too complex to be checked by hand

Two quotes by Jean-Pierre Serre (former Bourbaki member) in 1986

What shall one do with such theorems, if one has to use them? Accept them on faith? Probably. But it is not a very comfortable situation. (Talking about Feit-Thompson Theorem)

I am also uneasy with some topics, mainly in differential topology, where the author draws a complicated picture (in two dimensions), and asks you to accept it as a proof of something taking place in five dimensions or more. Only the experts can "see" whether such a proof is correct or not-if you can call this a proof

Interactive Proof Assistants: Why?

Complex mathematical proofs, such as the ones mentionned by Jean-Pierre Serre, are now checked in complete details with the help of interactive proof assistants

But maybe the main interest of such tools is that they can help the mathematicians in organising thoughts and in getting better *understanding*

One goal is to produce better proofs, keeping the connections with intuitions

(More in Johan Commelin's talk)

Interactive Proof Assistants: Why?

This provides new results in the mathematical study of proofs

New logical insight about the notion of *equality*, one of the most basic notion in mathematics, and unexpected connections with the field of *homotopy theory* (abstract study of "shapes")

Interactive Proof Assistants: How does it look like?

Development of proofs has strong analogy with development of programs

-same issues of modularity for large proofs and large software

-same issues of notations, choice of names

-same issues of maintenance: program repair/proof repair

-most developments are team work, like for large software developments

(This collaborative aspect was especially stressed by Voevodsky)

Interactive Proof Assistants: How does it look like?

So the development of formal proofs with a proof assistant is similar to the development of programs

But there is also an important *interactive* aspect

Some analogy with video games

Let me explain the analogy between proof assistants and video games. The main advantage of a proof assistant is that it gives the students immediate feedback. Incorrect proofs are rejected right away. Such rejections are an insult and challenge for motivated students. They will persist in their struggle to convince the machine. (Tobias Nipkow)

The Natural Number Game, version 1.3.3

By Kevin Buzzard and Mohammad Pedramfar.

What is this game?

Welcome to the natural number game -- a part-book part-game which shows the power of induction. Blue nodes on the graph are ones that you are ready to enter. Grey nodes you should stay away from -- a grey node turns blue when *all* nodes above it are complete. Green nodes are completed. (Actually you can try any level at any time, but you might not know enough to complete it if it's grey).

In this game, you get own version of the natural numbers, called `mynat`, in an interactive theorem prover called Lean. Your version of the natural numbers satisfies something called the principle of mathematical induction, and a couple of other things too (Peano's axioms). Unfortunately, nobody has proved any theorems about these natural numbers yet! For example, addition will be defined for you, but nobody has proved that `x + y = y + x` yet. This is your job. You're going to prove mathematical theorems using the Lean theorem prover. In other words, you're going to solve levels in a computer game.

You're going to prove these theorems using *tactics*. The introductory world, Tutorial World, will take you through some of these tactics. During your proofs, your "goal" (i.e. what you're supposed to be proving) will be displayed with a ⊢ symbol in front of it. If the top right hand box reports "Theorem Proved!", you have closed all the goals in the level and can move on to the next level in the world you're in. When you've finished a world, hit "main menu" in the top left to get back here.

For more info, see the FAQ.

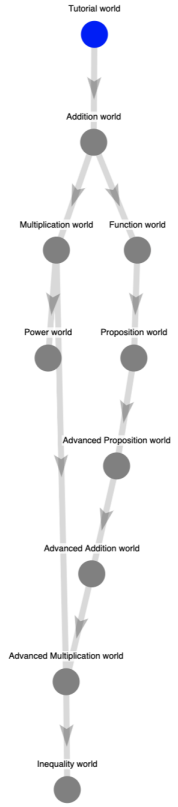
What's new in v1.3?

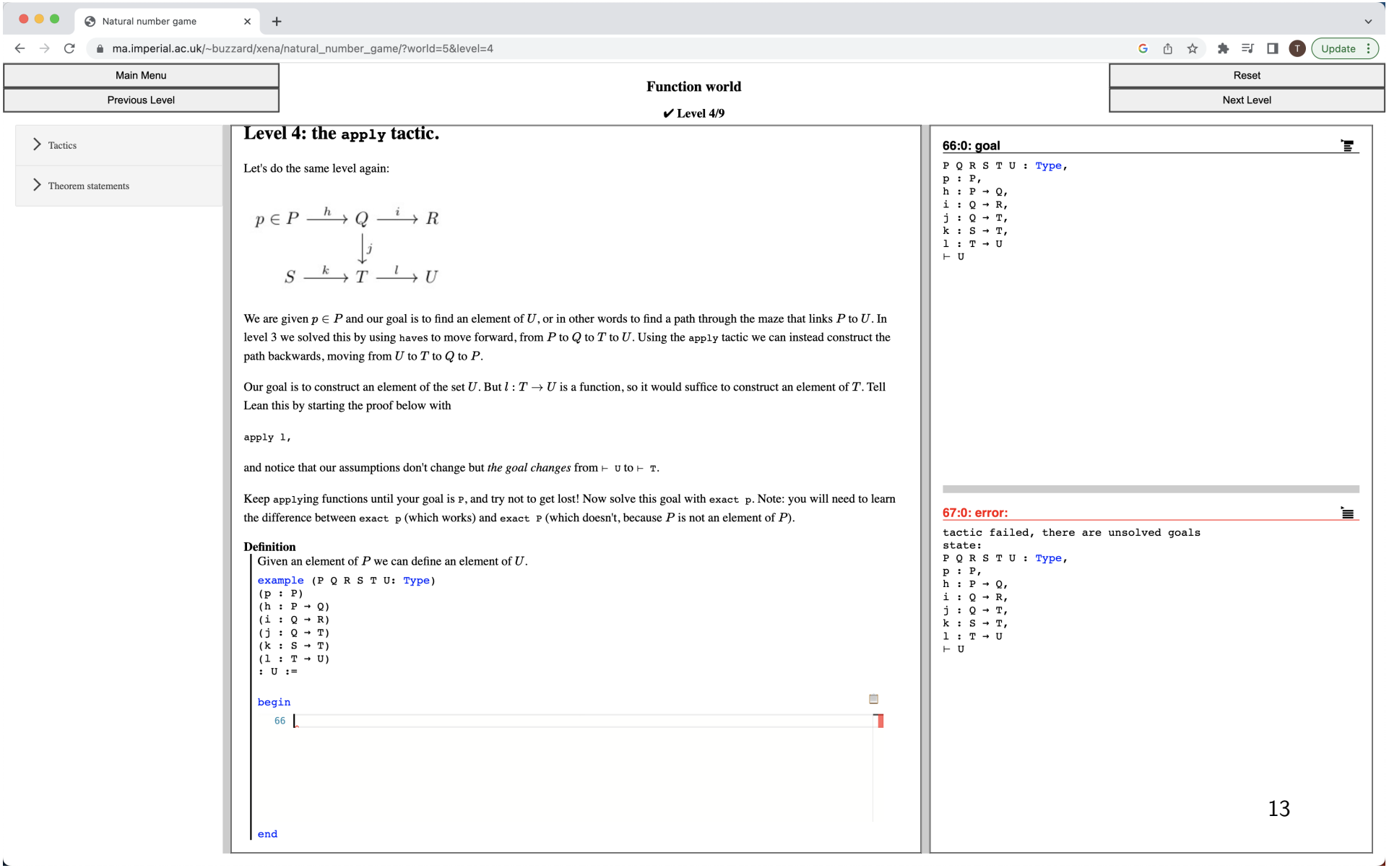
The game now saves your progress! Thanks to everyone who asked for it, and to Mohammad for making it happen :-)

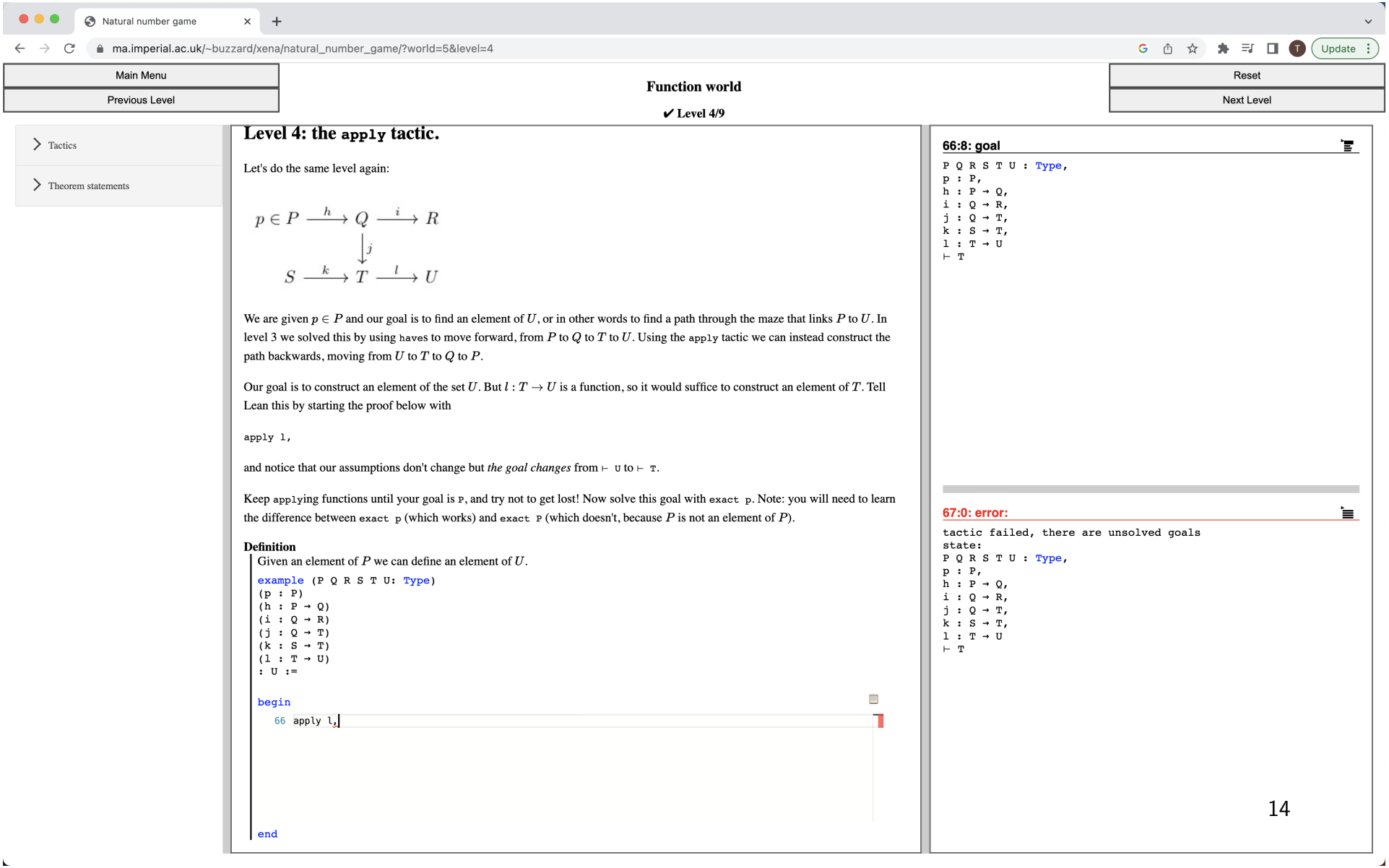
Cute little clipboard to copy your solutions.

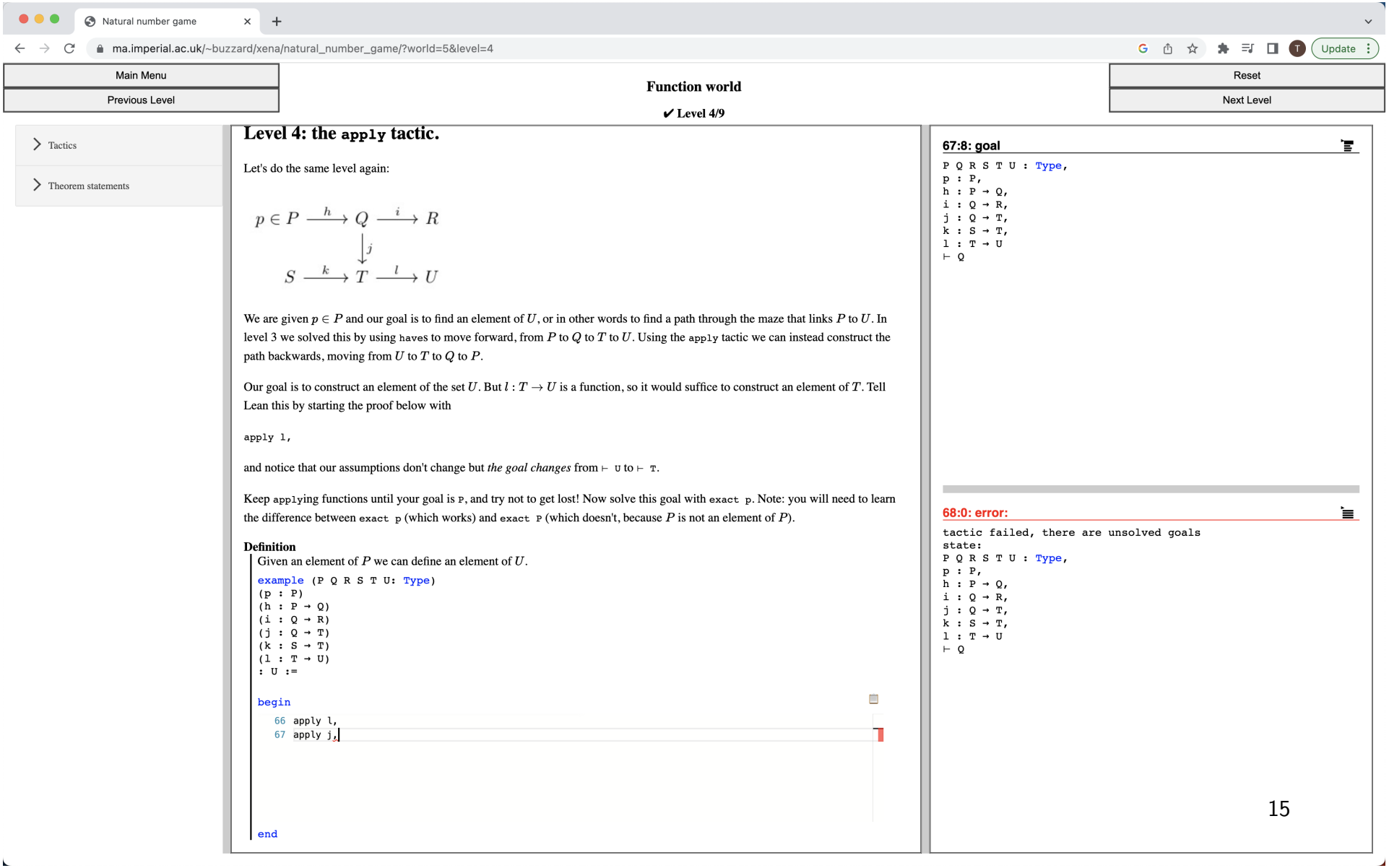
Thanks

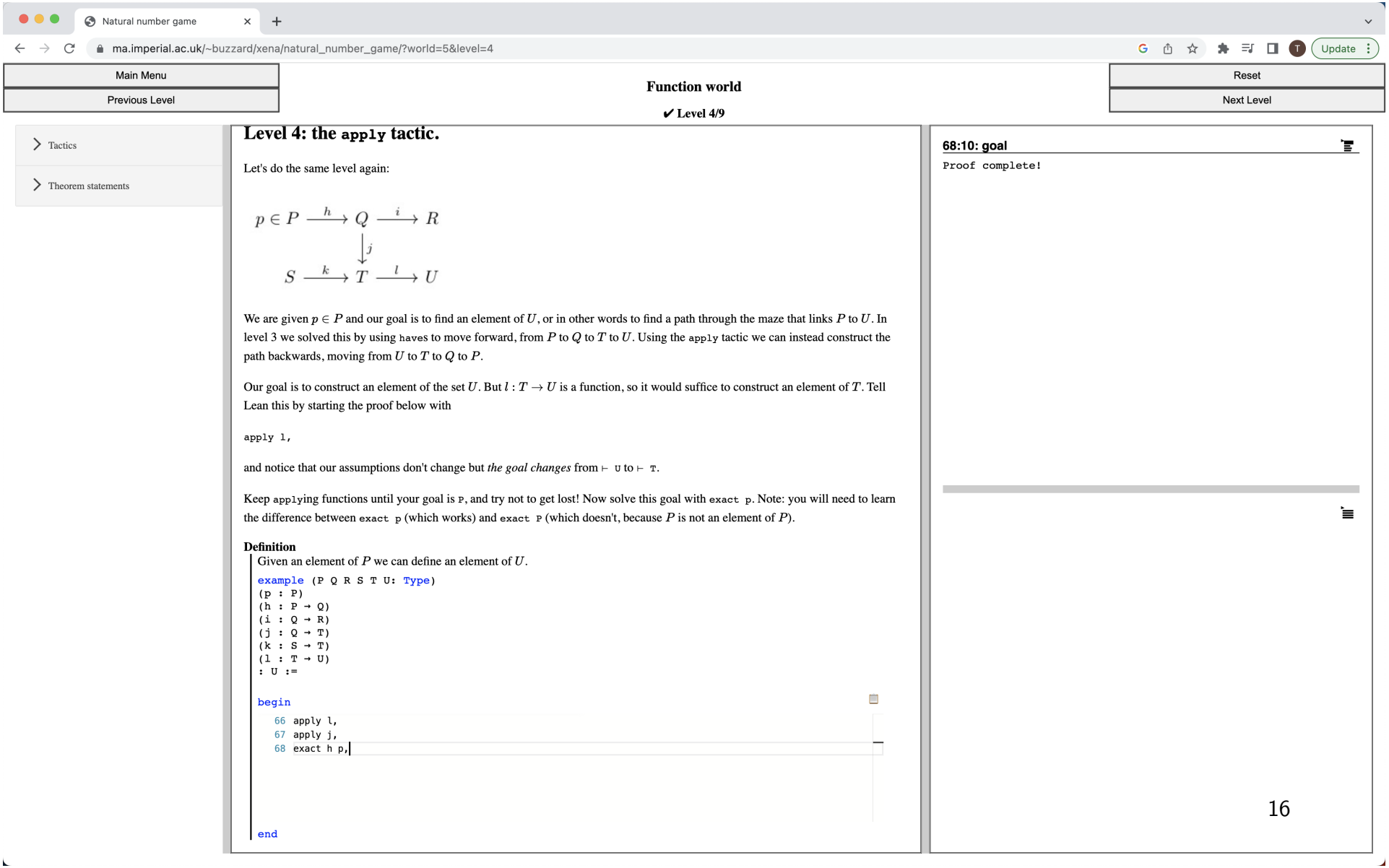
Special thanks to Rob Lewis for tactic hackery, Bryan Gin-Ge Chen for javascript hackery, Patrick Massot for his Lean to html formatter, Sian Carey for Power World, and, last but not least, all the people who fed back comments, including the 2019-20 Imperial College 1st year maths beta tester students, Marie-Amélie Lawn, Toby Gee, Joseph Myers, and all the people who have been in touch via the Lean Zulip chat or the Xena Project blog or via Twitter. The natural number game is brought to you by the Xena project, a project based at Imperial College London whose aim is to get mathematics undergraduates using computer theorem




Natural number game
ma.imperial.ac.uk/~buzzard/xena/natural_number_game/?world=5&level=4
Update
Main Menu
Previous Level
Function world
✔ Level 4/9
Reset
Next Level
Tactics
Theorem statements
Level 4: the apply tactic.
Let's do the same level again:
$p \in P \xrightarrow{h} Q \xrightarrow{i} R$
$\downarrow j$
$S \xrightarrow{k} T \xrightarrow{l} U$
We are given $p \in P$ and our goal is to find an element of U, or in other words to find a path through the maze that links P to U. In level 3 we solved this by using haves to move forward, from P to Q to T to U. Using the apply tactic we can instead construct the path backwards, moving from U to T to Q to P.
Our goal is to construct an element of the set U. But $l : T \to U$ is a function, so it would suffice to construct an element of T. Tell Lean this by starting the proof below with
apply l,
and notice that our assumptions don't change but the goal changes from ⊢ U to ⊢ T.
Keep applying functions until your goal is P, and try not to get lost! Now solve this goal with exact p. Note: you will need to learn the difference between exact p (which works) and exact P (which doesn't, because P is not an element of P).
Definition
Given an element of P we can define an element of U.
example (P Q R S T U: Type)
(p : P)
(h : P → Q)
(i : Q → R)
(j : Q → T)
(k : S → T)
(l : T → U)
: U :=
begin
66
end
66:0: goal
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ U
67:0: error:
tactic failed, there are unsolved goals
state:
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ U


Natural number game
ma.imperial.ac.uk/~buzzard/xena/natural_number_game/?world=5&level=4
Update
Main Menu
Previous Level
Function world
✔ Level 4/9
Reset
Next Level
Tactics
Theorem statements
Level 4: the apply tactic.
Let's do the same level again:
$p \in P \xrightarrow{h} Q \xrightarrow{i} R$
$\downarrow j$
$S \xrightarrow{k} T \xrightarrow{l} U$
We are given $p \in P$ and our goal is to find an element of U, or in other words to find a path through the maze that links P to U. In level 3 we solved this by using haves to move forward, from P to Q to T to U. Using the apply tactic we can instead construct the path backwards, moving from U to T to Q to P.
Our goal is to construct an element of the set U. But $l : T \to U$ is a function, so it would suffice to construct an element of T. Tell Lean this by starting the proof below with
apply l,
and notice that our assumptions don't change but the goal changes from ⊢ U to ⊢ T.
Keep applying functions until your goal is P, and try not to get lost! Now solve this goal with exact p. Note: you will need to learn the difference between exact p (which works) and exact P (which doesn't, because P is not an element of P).
Definition
Given an element of P we can define an element of U.
example (P Q R S T U: Type)
(p : P)
(h : P → Q)
(i : Q → R)
(j : Q → T)
(k : S → T)
(l : T → U)
: U :=
begin
66 apply l,
end
66:8: goal
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ T
67:0: error:
tactic failed, there are unsolved goals
state:
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ T


Natural number game
ma.imperial.ac.uk/~buzzard/xena/natural_number_game/?world=5&level=4
Update
Main Menu
Previous Level
Function world
✔ Level 4/9
Reset
Next Level
Tactics
Theorem statements
Level 4: the apply tactic.
Let's do the same level again:
$p \in P \xrightarrow{h} Q \xrightarrow{i} R$
$\downarrow j$
$S \xrightarrow{k} T \xrightarrow{l} U$
We are given $p \in P$ and our goal is to find an element of U, or in other words to find a path through the maze that links P to U. In level 3 we solved this by using haves to move forward, from P to Q to T to U. Using the apply tactic we can instead construct the path backwards, moving from U to T to Q to P.
Our goal is to construct an element of the set U. But $l : T \to U$ is a function, so it would suffice to construct an element of T. Tell Lean this by starting the proof below with
apply l,
and notice that our assumptions don't change but the goal changes from ⊢ U to ⊢ T.
Keep applying functions until your goal is P, and try not to get lost! Now solve this goal with exact p. Note: you will need to learn the difference between exact p (which works) and exact P (which doesn't, because P is not an element of P).
Definition
Given an element of P we can define an element of U.
example (P Q R S T U: Type)
(p : P)
(h : P → Q)
(i : Q → R)
(j : Q → T)
(k : S → T)
(l : T → U)
: U :=
begin
66 apply l,
67 apply j,
end
67:8: goal
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ Q
68:0: error:
tactic failed, there are unsolved goals
state:
P Q R S T U : Type,
p : P,
h : P → Q,
i : Q → R,
j : Q → T,
k : S → T,
l : T → U
⊢ Q


Natural number game
ma.imperial.ac.uk/~buzzard/xena/natural_number_game/?world=5&level=4
Update
Main Menu
Previous Level
Function world
✔ Level 4/9
Reset
Next Level
Tactics
Theorem statements
Level 4: the apply tactic.
Let's do the same level again:
$p \in P \xrightarrow{h} Q \xrightarrow{i} R$
$\downarrow j$
$S \xrightarrow{k} T \xrightarrow{l} U$
We are given $p \in P$ and our goal is to find an element of U, or in other words to find a path through the maze that links P to U. In level 3 we solved this by using haves to move forward, from P to Q to T to U. Using the apply tactic we can instead construct the path backwards, moving from U to T to Q to P.
Our goal is to construct an element of the set U. But $l : T \to U$ is a function, so it would suffice to construct an element of T. Tell Lean this by starting the proof below with
apply l,
and notice that our assumptions don't change but the goal changes from ⊢ U to ⊢ T.
Keep applying functions until your goal is P, and try not to get lost! Now solve this goal with exact p. Note: you will need to learn the difference between exact p (which works) and exact P (which doesn't, because P is not an element of P).
Definition
Given an element of P we can define an element of U.
example (P Q R S T U: Type)
(p : P)
(h : P → Q)
(i : Q → R)
(j : Q → T)
(k : S → T)
(l : T → U)
: U :=
begin
66 apply l,
67 apply j,
68 exact h p,
end
68:10: goal
Proof complete!

Interactive Proof Assistants: Can AI help?

For some part of mathematics, automatic tools (such as SAT, SMT solver, model checking) are very useful/essential

AI should also provide help in creative steps and intuitions

Creativity can be best expressed against a constrained/rigid framework and formal mathematics can be such a framework

AI can be incredibly good for some games; why not in the game of proving mathematical theorems?