

Isabelle Tutorial: Follow a Simple Theorem Step by Step in jEdit

G. Dalton Bentley

Contents

Chapter 1. Introduction	1
Chapter 2. Square Root of Two	4
§2.1. Background	4
§2.2. Implement in Isabelle	5
Chapter 3. Closing Thoughts	42
Appendix A.	43
Bibliography	45
Index	49

Introduction

Isabelle/HOL[1] is an interactive proof assistant using the Isar proof language, which combines English, logic and programming to read like a formal proof in mathematics. Other proof assistants include Coq, home website [6], compared with Isabelle in [7]).

A general definition of proof assistant might be [3]:

Proof assistants are computer programs that assist computer scientists, mathematicians and logicians in defining concepts and proving properties about them. Furthermore, they ensure correctness of the proofs that are constructed.

Isabelle, in its Isar incarnation, “is mostly concerned about proof documents, i.e. the real sources of some formal reasoning, in a form that is both human-readable and machine checkable.” [27]

HOL¹ is the Isabelle implementation of higher-order logic, “a polymorphic version of Church’s Simple Theory of Types . . . best understood as a simply-typed version of classical set theory.”[15] By “simple type theory” is meant that

Types can take types as parameters but not integers for example. The type of finite sequences (known as lists) takes the component type as a parameter, but there is no type of n -element lists. Logical predicates form the basis of a basic typed set theory, where any desired set can be expressed by

¹Isabelle provides several logic systems, with HOL being the best developed at this time. It includes an extensive library of concrete mathematics, as well as packages for concepts like co-inductive sets and types, and well-founded recursion. Sample proofs are provided.[15]

comprehension over a formula. We can define the set of n -element lists where the elements are drawn from some other set. Thus, the simple framework given by types can be refined through the use of sets.[14]

The process of writing a proof in Isabelle involves the user specifying each step (helped by the Isar extension of Isabelle that hides the complexity of the underlying ML implementation²), testing with one of many automatic proof methods, which also suggest applicable lemmas, and chaining the proven steps together.[3] As Wenzel puts it, this follows the tradition of “honest toil” as proposed by Bertrand Russell, with results derived from definitions of proper theorems (more than postulated axioms) in structured human-readable proof-documents (beyond mere machine scripts) relying on a strong type-safe ML logical core.[27]

The standard user interface for Isabelle is now jEdit, a GUI (graphical user interface) normally referred to as the PIDE, i.e., Prover Integrated Development Environment [4].

There are hundreds of pages of documentation provided with the Isabelle package download, but from our own experience we find that it is almost impossible to learn how to actually use Isabelle by simply reading them.³ Accordingly, we are going to take the approach here of presenting a simple mathematical proof and the interactions we made with Isabelle/HOL (we may refer to this application throughout as Isabelle or the jEdit user interface) to formalize this proof within the Isabelle context.

Why is there no tutorial included with the Isabelle system? There is one document included, *A Proof Assistant for Higher-Order Logic* [29], which is described as a tutorial, “a tutorial on how to use the theorem prover Isabelle/HOL as a specification and verification system” (to verify software, that being one of the initial motivations for developing automated proof systems, Intel having lost a half billion dollars over the 1994 bug in the Pentium floating point division unit[14]). However, the thrust of that article

²Scott’s Logic of Computable Functions (LCF) appeared in 1969. Around 1977 Robin Milner and his group developed Edinburgh LCF, a programmable proof checker. The meta-language of Edinburgh LCF, i.e., ML, provides not only a program execution language, but a general representation of logic in which inference rules are functions from theorems to theorems and theorems can be built only by applying validated rules to existing theorems.[17] Inference rules are represented literally as a list of premises and a conclusion. The basic idea of Isabelle is to construct proofs by unifying the conclusion of one rule with a premise of another.[18]

³Perhaps we have provided a hint of the complexity of the Isabelle system in the brief comments and footnotes thus far. We believe it is easier to learn to use Isabelle first, by going through some of the many proofs provided. If you want to learn more about the design and implementation later, the documents available will more than accommodate you, though that does require some ability and desire to form a wide and deep conception of the system (cross-referenced among 20 or more documents provided with the system, each giving pictures of different levels of abstraction and stages of development) as it evolved over the last three decades or more.

is functional programming rather than mathematical concepts, and is quite unusable as a general tutorial, as the authors acknowledge in their introduction, instead referring the reader to four additional documents (none of which are really tutorials either, but are helpful in understanding the system once the user has gotten his feet wet, so to speak). The best tutorial we have encountered was “aimed at mathematicians who would like to use it for the formalization of mathematical results,” [22] though slightly out-of-date and not using the current interface jEdit (nevertheless, very usable).

Our goal will be to provide a tutorial that walks a newcomer to Isabelle/jEdit step by step through the construction of a simple mathematical proof, providing frequent screen shots and discussion to assure the reader is on board. The reader is strongly encouraged to enter and execute the theory lines in an open jEdit/Isabelle session, there being no substitute for learning by doing in this context. We will provide frequent cites to relevant Isabelle documentation titles and sections in an attempt to provide somewhat of a topic index into the twenty or more portable document format references accompanying the distribution (as we noted above, this system is wide and deep). We include cites to a number of external papers by the Isabelle developers also.

Square Root of Two

2.1. Background

Our test case here is concerned with that venerable question, “Can one find a rational number whose square is precisely the number 2?” This was the issue that supposedly caused much upset to the Pythagoreans¹:

Recall that a fraction—that is, a rational number—is something that can be expressed as the ratio a/b of two integers (or whole numbers) a and b , with b non-zero . . . The Pythagoreans had originally hoped that all their geometry could be expressed in terms of lengths that could be measured in terms of rational numbers . . . If all geometry could be done with rationals, then this would make things relatively simple and easily comprehensible. The notion of an “irrational” number, on the other hand, requires infinite processes, and this had presented considerable difficulties for the ancients (and with good reason).[5]:

From the Pythagorean theorem $2b^2 = a^2$, a square (in Euclidean geometry) with unity side length $b = 1$ would have a diagonal of $a = \sqrt{2}$.

¹Burgess points out that we do not have much information about Pythagoras, other than comments by those writing generations following his death.[10]. However, it is generally agreed that “[Greek] geometry developed itself for some time on the basis of the numerical proportion which was inapplicable to any but commensurable magnitudes, and that it received an unexpected blow later by reason of the discovery of the irrational.”[11] See also [12]. We mean by lengths a and b “commensurable” that a common measure, e.g., a ruler, could be positioned end-to-end m times to measure length a exactly and n times for length b , where m, n are whole numbers and the resulting ratio is $a:b = m:n$ [10].

The problem is usually then stated as “If $\sqrt{2}$ is rational, then the equation $a^2 = 2b^2$ is soluble in integers a, b with $\gcd(a, b) = 1$ ”².

We will prove that the square root of any prime number (where a prime number is a positive integer greater than 1 whose only factors are 1 and itself³ is irrational and then simply note that 2 is a prime number, making its square root irrational. We follow one of the Isabelle/HOL theory files (with file extension `.thy`) that accompany the Isabelle distribution, i.e., `HOL/ex/Sqrt.thy`, which contains several proofs, each using somewhat different Isabelle methods. The mathematical (in human language) statement of our theorem and proof is:

Theorem 2.1. *If p is prime, i.e., $p \in \mathbb{Z}_{>0}$ with $p > 1$ and having only factors 1 and itself, then $\sqrt{p} \notin \mathbb{Q}$, where $\mathbb{Q} = \{\frac{a}{b} \mid a \in \mathbb{Z} \text{ and } b \in \mathbb{Z} \text{ and } b \neq 0\}$ and there is a unique (reduced) representation such that $\gcd(a, b) = 1$, i.e., (a, b) are relatively prime:*

Proof. Assume on the contrary that $\sqrt{p} \in \mathbb{Q}$: Then $\exists m, n \in \mathbb{Z}$ with $n \neq 0$ such that $\sqrt{p} = m/n$ with $\gcd(m, n) = 1$, i.e., (m, n) are relatively prime. In that case, $m = |\sqrt{p}|n$. It follows that $m^2 = (\sqrt{p})^2 n^2$ and $m^2 = pn^2$. In that case we may say that $p|m^2$ and also that $p|m$, since a prime divisor of a product of integers must divide one of the integer factors, which are identical in the case of a square. That being so, there must be an integer k such that $m = pk$. Since we established that $m^2 = pn^2$ and we can square $m = pk$ to obtain $m^2 = p^2 k^2$, we have $pn^2 = p^2 k^2$. We can divide that last result by p to obtain $n^2 = pk^2$. That tells us that $p|n^2$ and therefore $p|n$ (as above, prime divisor of product of integers divides at least one of the factors). We have therefore shown that p prime divides both m and n and therefore must divide the greatest common divisor of m, n , i.e., $p|\gcd(m, n)$. However, we stipulated that m, n are relatively prime, i.e., that $\gcd(m, n) = 1$. In that case, $p = 1$, a contradiction since we stated p is prime and must therefore be greater than 1, i.e., $p > 1$ by definition of primality. Our assumption that $\sqrt{p} \in \mathbb{Q}$ is therefore false and therefore $\sqrt{p} \notin \mathbb{Q}$. \square

2.2. Implement in Isabelle

Let us now develop the `SqrtTutor1.thy` theory Isabelle implementation of the proof 2.1 above for Theorem 2.1, discussing the output state and context for each step in the automation of the proof. We will include various screen shots (which should be of sufficient resolution that you can zoom in your

²Edited from Theorem 43 in Hardy and Wright *An Introduction to the Theory of Numbers* as quoted in [13]

³For our context, where irreducibility and primality occur together in the positive non-zero subset of \mathbb{Z} we can also say that if a prime divides a product then it always divides one of the factors of the product.[13]

pdf viewer to see detail) and text from Isabelle/jEdit. As we mentioned earlier, we may refer to jEdit or Isabelle depending on whether we refer specifically to Isabelle processing or procedure or whether to direct input to, or observer output from, the jEdit interface window. If you are using a previous or later edition of Isabelle you may encounter slight differences in syntax (which jEdit will nag you to fix) or display.

When you start the Isabelle application, by default you will be editing the empty `Scratch.thy` theory 1 (the image resolution is sufficient to permit zooming in your viewing application for detail). Rename that to `SqrtTutor1.thy` and **Save** or **Save as**, the latter option giving you the opportunity to note where the file is being saved or change that location.

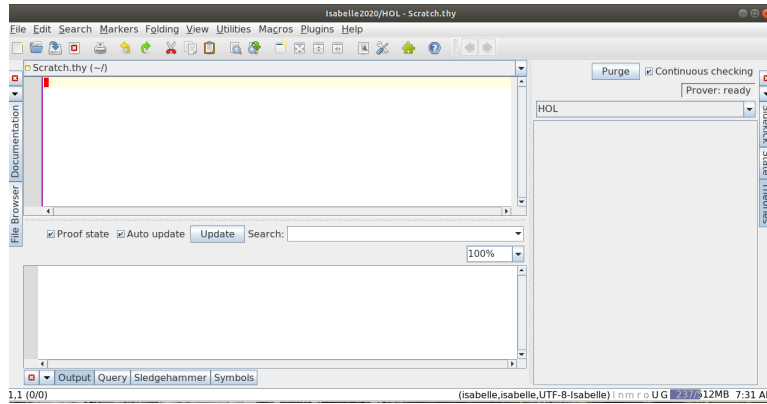


Figure 1. Initial screen upon opening Isabelle/jEdit.

The large central pane⁴ in the jEdit window is referred to as “the main text area” in the *Isabelle/jEdit* document[23] by Makarius Wenzel that is included with the Isabelle distribution package, filename `jedit.pdf` (we recommend you keep that document handy). We will refer to the main text area as simply “the edit area” (or the edit pane). For Isabelle to recognize our work as a theory we must include a line in the theory file associating the overall theory document with the name of the file itself (without extension). Accordingly, type in the edit area:

```
theory SqrtTutor1
imports Complex_Main "HOL-Computational_Algebra.Primes"
begin
```

⁴Throughout the official documentation for jEdit, [23] and in §2.2 in particular, these subsections of the overall jEdit graphic user interface screen real estate are referred to as *panels* when docked (and *floating windows* if undocked). We prefer the term *pane*, perhaps because of the more direct window connotation, i.e., the view port for a particular functional division of the jEdit application (“you say *potāto*, I say *potāto*”).

```
theorem
```

```
end
```

The `imports` command reads in two background theories that we will require in our own theory. We are cheating here in that we already knew we would need these two theories. It could be that you were beginning entirely *de novo* and would have to search for additional background theory support you need to construct your own proof. The amount of preloaded support that Isabelle includes is prodigious and that may also be supplemented by the Archive of Formal Proofs. On a Linux machine it is fairly easy to rapidly search for theory filename patterns or even to search in a few seconds all of the text of the thousands of theories for terms of interest, e.g., `::nat` if you were interested in definitions and properties declared regarding the type of natural numbers in Isabelle. On a Windows box you can still search theory filenames and jEdit itself can search its current *session*.

From the Isabelle document *The Isabelle System Manual*⁵, Chapter 2, we are told that an Isabelle *session* consists of a collection of theories in a hierarchy based on a major object logic like HOL (itself built on a combination of `Pure` and the ML of the kernel). This provides the background that is required for formulating statements and composing proofs, acting as a medium to produce formal content, depending on earlier material (declarations, results etc.), paraphrasing from another Isabelle document, *The Isabelle/Isar Implementation*, filename `implementation.pdf`.^[24]

The HOL session is pre-built as a persistent heap image that loads on invocation of Isabelle (unless you choose to work in another of the logic modules included with Isabelle, e.g., ZF or Zermelo-Fraenkel Set Theory). To build such a session from scratch takes some time, even given with heavily parallel processing used by Isabelle code, on a machine with 8 cores and 8 GB of RAM (considered as a “small” machine environment in the Isabelle context; we can testify that we barely get by with 8 cpu threads on 4 physical cores and 4 GB of RAM running Linux). As we said though, HOL is already built and our `imports` line above only brings in a few more theories with material pertinent to our own application here.

This `theory-imports-begin-theorem-end` block you typed in above constitutes the top level specification for our theory.⁶ We emphasize again

⁵Provided as `file system.pdf` in the Isabelle distribution.^[25]

⁶Each `theory` container is derived from a certain sub-graph of ancestor theories. The `begin` operation starts a new theory by importing and merging contents of parent theories, entering incremental update mode until the final `end` operation is performed. All of this occurs in the internally bootstrapped Isabelle logical context acting as the background required to formulate statements and compose proofs. Derivations within this logic context can be described as a

that the name following `theory` must be the *actual name of the file on the computer that you save the statements to* (Figure 2). That is `SqrtTutor1.thy` (that is a 1, not a lower case “l”, because this is the first tutorial) in the instant case, but do not include the suffix `.thy`. Do include the suffix in the computer file name when you **Save**. We are belaboring this point here because inattention to detail can cause jEdit to refuse to cooperate with you, and little is as frustrating as being unable to follow even the initial steps of a tutorial (which might cause some to simply discard the software).

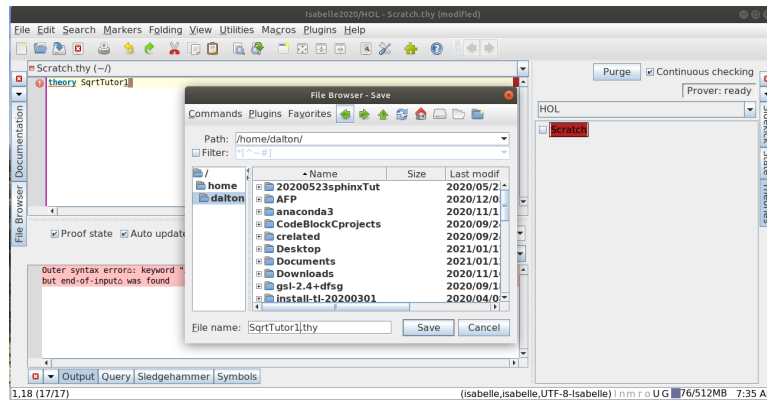


Figure 2. Saving the theory file under new name, being mindful of path.

Similarly, assure that you place the `imports` line prior to `begin` above. We technically enter proof mode with the statement `theorem`. That could include a name if we planned to refer to it later⁷, e.g.,

```
theorem irrat_sqrt_prime.
```

You may find jEdit displaying complaints now in the Output pane below the edit pane rather than an acknowledgment of the proof state:

```
Outer syntax error: proposition expected,
but end-of-input was found
```

We note that there is a little “up spade” symbol in that output above, but we do not have the Isabelle fonts⁸.

judgment $\Gamma \vdash_{\Theta} \varphi$, meaning that proposition φ is derivable from hypotheses Γ within the theory Θ .^[26]

⁷jEdit/Isabelle should notify you if it recognizes that name as already in use, but it is probably better not to name experimental work not intended to be used by other theories.

⁸Isabelle can print \LaTeX typeset pdf, in fact most of the system documentation was produced in Isabelle, but it is an incredible chore at this time, involving batch modes, special session builds that can take off unexpectedly, directory and file creation and movement and adjustment; they hope to make this a simple one-button task from jEdit eventually. In the meantime, jEdit will print a usable browser preview of the current main edit buffer using Unicode characters, which we save as `html` or print to pdf from the browser, e.g., **Chromium**, and find useful for later reference.

The downside of the continuous proof checking with asynchronous user interaction⁹ is that you must immediately learn to ignore the jEdit complaints, whether in the Output pane below the edit area, or manifested as red fonts or other signals in the edit area, until you have completed multiple commands or token entry. You can disable this continuous backseat driving by jEdit by unchecking **Auto update**, but it is very useful (when you are ready for it) and also efficiently uses cpu resources. We keep the **Proof state** checkbox checked also, preferring to see that state in the Output pane also, but unchecking it relegates that information to one of the side panes on the right of jEdit, where you can instead choose one of those tabs to look at it when you are ready.

In the present circumstances, jEdit knows that additional commands or parameters should follow, in particular that we should provide a proposition (per Output pane message above), having set up a theory block. We note that jEdit highlights (in red) questionable syntax in the edit area. The individual tokens in the edit area will appear blue, green, or other less angry colors¹⁰ once they are accepted in the continuous parsing and checking underway.

Let us now follow the proof 2.1 above for Theorem 2.1, making our initial assumption and plan known to Isabelle by typing in the edit area of jEdit, following **theorem**

```
assumes "prime (p::nat)"
shows "sqrt p \<notin> \<rat>"
```

Notice that as you type in the **assumes** proof command of the top-level Isar engine responsible for theory and proof elements, jEdit colors it green to indicate to you that it understands what you are trying to do (and the **theorem** token previously typed above should be colored blue once jEdit has enough input to approve of your structure). For the quoted expression following, we usually type both quotation marks for the inner syntax Isabelle types and terms expressions¹¹ together, e.g., “”, and then fill in the expression to be quoted, here **prime (p::nat)**.

With the first line of the last typed input, we are telling Isabelle that we are assuming the variable **p** to be a prime number of type **nat** (type two

⁹The Isabelle prover works in parallel on multiple cpu cores, providing feedback in the jEdit output pane and edit area, while accepting new user input asynchronously.[23]

¹⁰Depending on the revision date of Isabelle, the operating system environment, and possibly your own customizing, the colors may not be exactly identical to ours.

¹¹They are not strings, but expressions of Isabelle types and terms of the logic, so-called *inner syntax* which are combined with the theory source commands of Isar (*outer syntax*), i.e., the specifications and proofs. Chapter 3 of *The Isabelle/Isar Reference Manual*[24] discusses this in more detail, but we recommend to pursue detail only after obtaining some familiarity in use here.

colons to connect the type with the variable name) for natural number.¹² You should see jEdit indenting the statement a few spaces from the outer **theorem** line automatically to make the block structure clear. After hitting the **Enter** key on the keyboard, we typed the **shows** line above, which should also be colored green and indented to the same level as **assumes**.

After you typed **sqrt p** you should have observed **p** turn blue to indicate that Isabelle recognizes it as a free variable (which was declared on previous line to be a prime number). If you are a touch-typist, you should consciously slow down when beginning to type a symbol like the `\<notin>` (referring to the relation “not in set” or “not a member of”) or `\<rat>` for rational numbers \mathbb{Q} . Looking at the screen shot, you see after we have typed `\<not` or so, jEdit pops up a tooltip box with suggestions as to what symbol(s) you might want here (zoom your display of this tutorial if necessary), also complaining of malformed command syntax through the Output pane below.

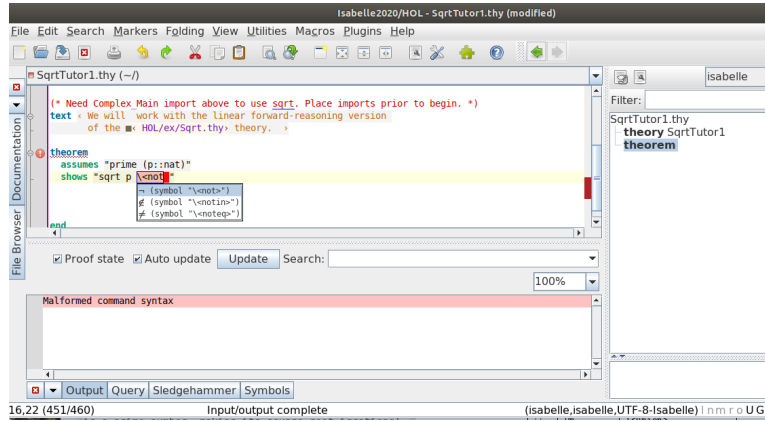


Figure 3. Popup menu offers to help insert commands.

You can left-click (or equivalent for your interface device) one of the choices in the popup menu with the mouse or buttons on your touchpad, or you may use down-arrow (on your keyboard) to scroll down to the next possibility to highlight it in blue. You can hit the **Tab** key on your keyboard to cause that selection (it will be highlighted blue if it is selected) to be inserted at that point in the edit window (if you did not already left-click with mouse). You obtain the same assistance when typing the code `\<rat>` for the \mathbb{Q} symbol for the set of rational numbers.

If you do not know the code for the desired symbol, you can use the **Symbols** tab at the bottom of the jEdit window, switching the lower pane

¹²We specified positive integer in our earlier proof statement, but this is synonymous and easier in this context.

display from Output to Symbols. Then select what kind of symbol you are after, “Letter” in this case. Scroll down to the line with the blackboard outline \mathbb{Q} and hover your cursor above that key on the display to have the code pop up. At this point you can simply left mouse click that key and jEdit will insert the symbol at the current position in the Edit pane above. As you begin to recall the codes (from repeated encounters in your Isabelle work) you can skip this and just type enough of the code to obtain the suggestion popup as we described above.

The Output pane (select that tab at the bottom of jEdit pane where you were looking at Symbols) in jEdit may show various complaints while you finish typing, but if you satisfy the syntactic and logical requirements, the complaints (and pink highlighting) should disappear, to be replaced, in the present context, by the proof state (which is “prove”):

```
proof (prove)
goal (1 subgoal) :
1. (sqrt p)  $\notin \mathbb{Q}$ 
```

You could also see the current proof state (the information above) of the Isabelle/Isar engine by selecting the right pane tab “State.” Because we have selected the checkbox for proof state, the Output pane below our edit area also displays this proof state information along with error messages. Perhaps you would prefer to see only error messages in the low pane and look to the right to see state (we prefer state and error message together in the Output pane).

You can also select Sidekick mode in that right pane, which shows an outline of the theory document you are entering. When you initially type in the `imports` statement discussed above, you can go to this right pane and select the “Theories” tab to see the theories that Isabelle loads as a result and the progress of that load (see screen shot 4, which includes some lines we have not typed yet¹³). There is so much information available in this jEdit appliance that you really have to treat it like the instrument panel in an airplane cockpit, scanning around the available indicators for changes or exceptions and focusing on indicators connected with your immediate actions and needs.

Isabelle has informed us we are in *prove* state, so let us type `proof` on the next line in the edit area and hit carriage return. The Output pane

¹³We remind you that the images are of sufficient resolution that you may zoom in with your pdf viewer to see details.

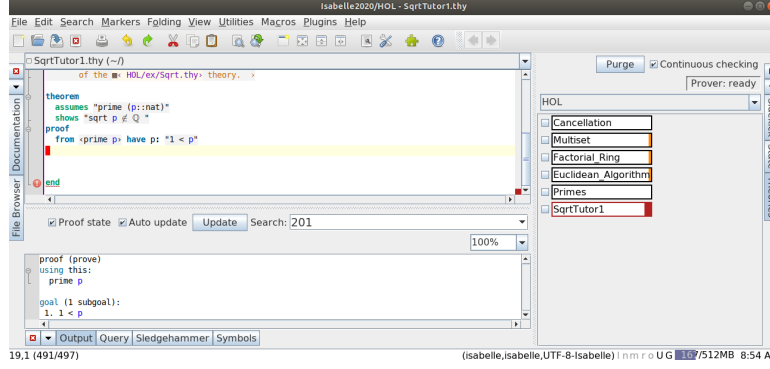


Figure 4. See Theory load status in right pane with Theories tab selected.

proof status changes to

```
proof (state)
goal (1 subgoal) :
1. (sqrt real p) ∈ ℚ ⇒ False
```

In other words we are now in *state* mode, where Isabelle/Isar is ready to accept declarations and claims supporting our goal, the statements we want to prove. The Isabelle/Isar system maintains a proof state, i.e., a set of premises and a set of goals, the statements to be proved. The final proof will set out the sequence of inference rules, that are applied to the proof state to reach the goals. The Output pane text tells us that we are working on the primary (outer indent level in the theorem block structure) goal, i.e., to prove $\neg (\forall p(\text{prime})(\sqrt{p} \in \mathbb{Q}))$.¹⁴

Following our original statement above for Theorem 2.1, let us establish the fact that if p is prime, then p should be greater than 1, 1 of course not being prime by definition (this fact will be used in the proof, but was stated in the theorem 2.1 proper earlier). We type on the line following **proof**, the token **from** and then, slowly as before to give jEdit time to pop up a suggestion, `\<op`, whereupon jEdit pops up a tool tip suggesting `< (symbol "\<open>")`. This is what we want, so hit the Tab key or click on the suggestion with the mouse to insert it. Now type **prime** (no quotes), space, followed by **p**, then begin to close the term with `\<c1` and jEdit will

¹⁴Why does the Output pane message imply that Isabelle believes our variable p is real? The square root operation in Isabelle/HOL is defined only for real numbers to our variable is temporarily coerced to that type. You might view numeric types forming an inclusion chain $\text{nat} \sqsubseteq \text{int} \sqsubseteq \text{rat} \sqsubseteq \text{real} \sqsubseteq \text{complex}$ where types on the left can be transferred to the type on the right with the corresponding characteristic algebraic structures.[8] See also [24] discussion of coercive subtyping, [9] definitions for overloaded functions, and a comprehensive examination of the subject in [16]. We discuss this issue again farther below.

suggest `\<close>`, which is correct, so Tab or click on it to insert the closing angle around `prime`. Note that quotes and cartouche angle bracketing are synonymous in this context, but you must use quotes around module imports like `HOL-Computational_Algebra.Primes` at the top of our theory document (`Computational_Algebra` is a module, i.e., in a subdirectory within `HOL`, while `Primes` is a theory found there and has a suffix `.thy` on the disk).

What we are doing with `from prime p` is telling Isabelle to take note of the fact that p is prime and then prove the inference (which we will type in shortly) that p must therefore be greater than 1.¹⁵

jEdit should indent `from` and remainder of the statement, coloring `from` and our free variable `p` blue. With cursor now just following `<prime p >`, the Output pane now tells us:

```
proof (chain)
picking this :
prime p
```

Isabelle/Isar is telling us that we are now in an intermediate state (`chain`) between proof (`state`) and proof (`prove`), that is, the immediate claimed fact (i.e., that `p` is `prime`) has just been picked up in order to support an expected claim. See Figure 5 (zoom for detail), which provides a visual interpretation of processing the Isar proof language (our typed input) as transitions of the Isar virtual machine, similar to the evaluation of algebraic expressions on a stack machine (figure and description from §2.2.3 of the *Isar Reference* [24]).¹⁶

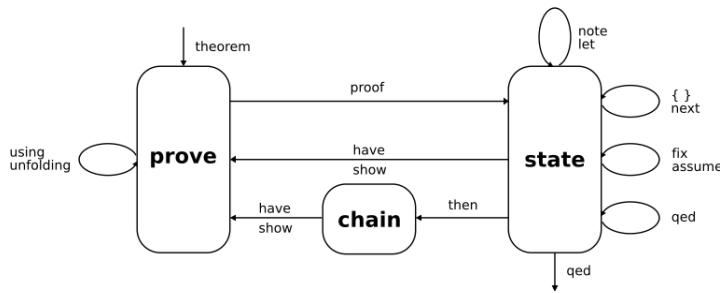


Figure 5. Modes of the Isar virtual machine.

¹⁵The keyword `from` is an abbreviation for `note a then`, i.e., note this fact a then proceed to the inference we would like to draw as a new fact in our collection.

¹⁶You may want to refer back to this diagram whenever we mention state transitions in order to build a conception of our interaction with Isabelle.

Our claim is that “if p is prime then 1 is less than p ,” so type: `have` p : “ $1 < p$ ” (that less-than symbol $<$ can be typed with standard keyboard; enclose “ $1 < p$ ” in quotations or cartouche it with `\<open>\<close>` as we did above) to complete the line. The p : portion is simply a label so we may refer to this fact later on in our proof once the fact is established. Your jEdit window should look like this screen shot 6.

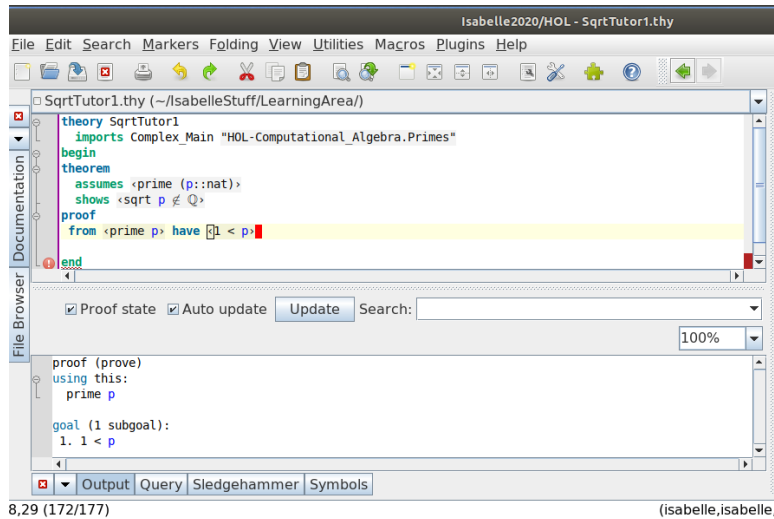


Figure 6. Status of jEdit session at line 8.

Note that the Output pane has updated to:

```

proof (prove)
using this :
prime p :
goal (1 subgoal) :
1. 1 < p

```

Isabelle/Isar is informing us that it has transitioned to **prove** mode using the fact that p is **prime**, with the subgoal of proving that necessarily $1 < p$. How do we prove this? We know by the definition of a prime number that this is true if p is prime, but we do not know how Isabelle/Isar characterizes that fact. If we switch the lower pane from Output to Query and select Print Context (with context, terms and theorems checkboxes checked, and hit the Apply button to update that pane), we see that the `?thesis` is now $1 < p$, i.e., that is now the innermost chain conclusion we want to prove (refer to jEdit window screen shot 7). Again, these screen shots have sufficient

resolution that you may zoom your pdf viewing application to see details clearly.

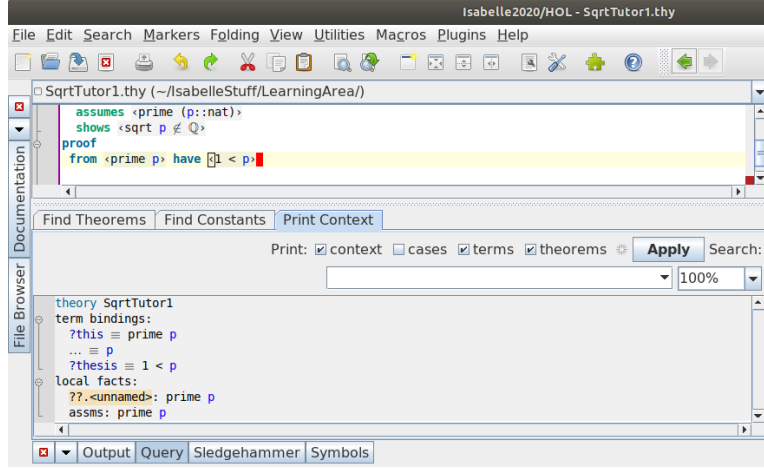


Figure 7. Context at line 8 awaiting proof $1 < p$.

There are a couple of paths we can take (short of text searching the entire HOL folder on the computer for term “prime” or the like). If we return the bottom pane to Output tab and type `solve_direct` on the next line in the edit area, Isabelle checks whether the current subgoals can be solved directly by an existing theorem (the theories available in our context are shown in the right pane¹⁷, Theories tab, one of which, for example, is Primes). In the Output pane after we type `solve_direct`, we see 8)

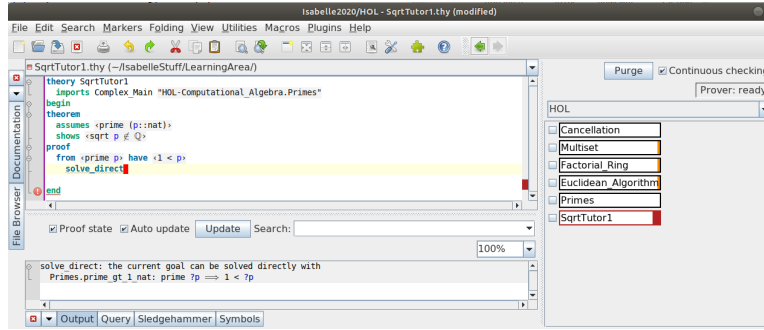


Figure 8. solve direct suggestion for proving $1 < p$.

The Output pane suggestion is that we might be able to prove our subgoal using a rule, `prime_gt_1_nat`, in the Primes theory file (notice the

¹⁷Those theories supplement the large body of theories loaded on invocation of Isabelle in the heap image for HOL.

Theoryfile.rule format of the suggestion).¹⁸ Another way to obtain that solution suggestion would be to erase the `solve_direct` line in the edit area, then switch from the Output to the Query tab on the lower pane. Then select the Find Theorems tab and type `solves` in the Find entry field and click on Apply. That also suggests the `prime_gt_1_nat` rule, as you can see in the screen shot 9

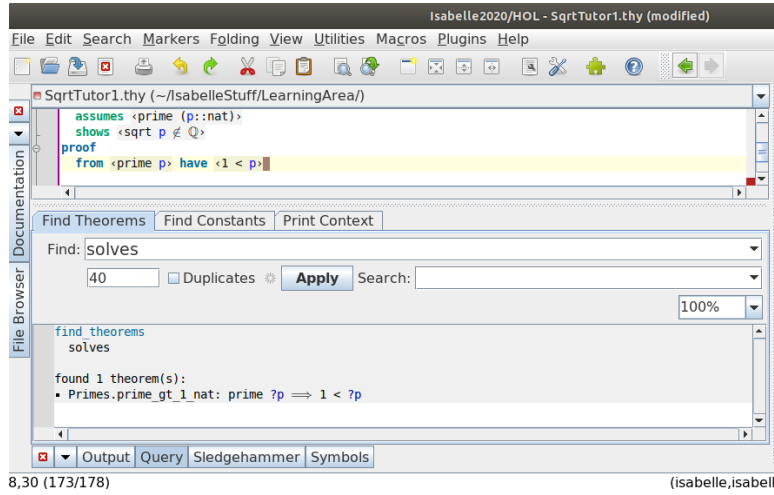


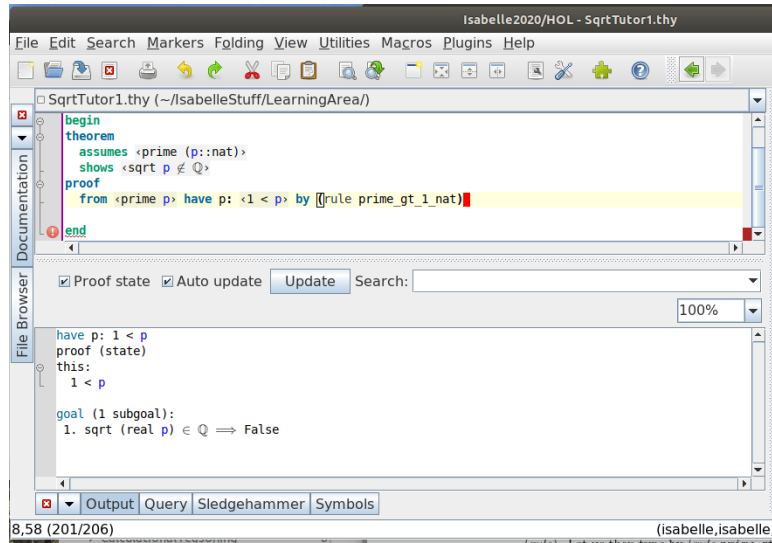
Figure 9. Query Find suggestion for proving $1 < p$.

So how do we tell Isabelle to attempt to solve the current subgoal using this rule? In human natural language we could say we would like to prove that the prime is > 1 “by the rule” `prime_gt_1_nat`. In §2.3.2 in the *Isar Reference* [24] we see examples using that idiom `have by (rule)`. Let us then type `by (rule prime_gt_1_nat)` at the end of our current line in the edit area. Looking at the screen shot 10, we see the Output pane now acknowledges we have $(1 < p)$ available as a fact¹⁹ and the goal state reverts to the outer block primary goal, $\sqrt{p} \in \mathbb{Q} \implies \text{False}$:

It is worth digressing for a moment to look at the source of the lemma we just used to establish our fact that $1 < p$. If you click on the File Browser tab on the left hand side of the jEdit window, you will open that pane and obtain

¹⁸For your information, the long outline arrow \implies in the Output pane description of what the suggested rule could do for us is part of the Isabelle framework, structuring theorems and proof states, separating assumptions from conclusions as in $[A_1; \dots; A_n] \implies A$, or in the present case, object `?p` (the question mark denotes schematic variable that is populated per the logical context) with property `prime` implies the property $(1 < ?p)$. See §2.4.1 in [26] and Chapter 3, and §10.3 in [20] (that entire paper is useful). It is not the implication \longrightarrow symbol that is part of the HOL logic (see §2.1.1 in [28]).

¹⁹You could also examine the local facts using the lower pane Query tab and clicking on Apply in Print Context to refresh that display. The local facts listed are now both that p is prime and that $1 < p$.

Figure 10. Prove prime implies $1 < p$ by rule.

(to the left of the edit area in jEdit) a Path field, below that a hierarchy of folders below the Path, and below that still another unfolding pane where you can open the selected folder to locate the theory file of interest (which we double-clicked to open) ¹¹

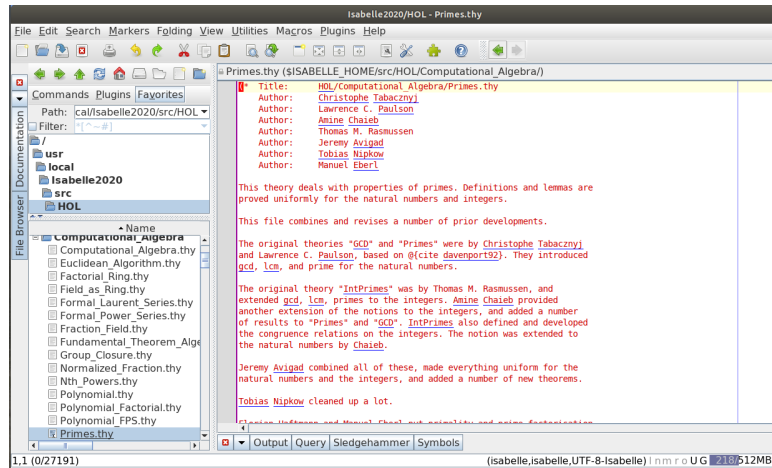


Figure 11. File Browsing in jEdit to examine the Primes.thy file.

Note that we temporarily closed²⁰ the lower pane in jEdit (you can still see the lower pane Tabs at the bottom of the screen), as well as the right

²⁰All of these panes close if the active tab is clicked, in this case Output. Once they are closed, you can open them again by another click.

hand pane (where you could still see the tabs of that pane at the right except for the fact that we cropped the image) in order to provide more space for the display of the `Primes.thy` file. If you scroll down the `Primes.thy` file in the edit area you can find the `prime_gt_1_nat` lemma 12. You can see that this lemma in turn is proved using another lemma, `prime_ge_2_nat`, proven earlier in the `Primes.thy` file.²¹

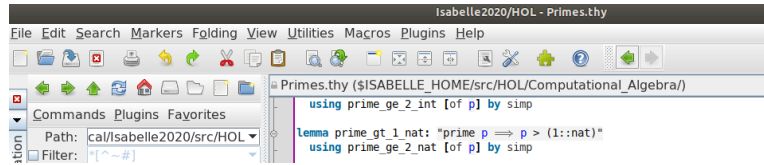


Figure 12. Scroll down `Primes.thy` file to lemma of interest.

Let us return to our work on the `SqrtTutor1.thy` theory. Go to jEdit File→Close global to close `Primes.thy` and delete it from the jEdit buffers, at which time our `SqrtTutor1.thy` will reappear in the edit area. Then click on the left hand File Browser tab to close that pane. Then click on the Output tab at the bottom of the jEdit window to open the Output pane again. Finally, click on the right hand tab Theories to open that right hand pane again. We hit the **Enter** key on the keyboard to position the cursor at the next line in the edit area.

Proceeding with the proof 2.1, we will assume that, contrary to our thesis, $\sqrt{p} \in \mathbb{Q}$. Type in the edit area `assume "sqrt p ∈ ℚ"`.²²

Type on the following line then obtain `m n :: nat where`. Then type²³

```
n: "n \<noteq> 0" and sqrt_rat: "\<bar> sqrt p \<bar> = m / n"
```

We gave you the command code for the absolute value bars in that last line. You cannot simply type the absolute value | vertical bars (to surround `sqrt p`) on the keyboard. You can also go to the Symbols tab on the lower pane (below the edit pane) and select the Punctuation tab, where you will find the broken vertical bar. You can click on it to insert the symbol at the present cursor location in the edit pane, or you can take note of the code displayed when you hover cursor on the button (and then type that in as we did). Similarly, instead of the code for \neq we typed, you could go to

²¹We mentioned in the Introduction 1 that this is the LCF way, i.e., inference rules are functions from *validated* theorems to theorems.

²²You may need to return to the first few pages of this section 2.2 where we discuss quoting, cartouches, tooltips, etc. if you have a problem here. The index following this tutorial document proper is hyperlinked by term to the relevant pages, which might be another way to quickly refresh your memory of details.

²³We are following our proof 2.1, knowing that if $\sqrt{p} \in \mathbb{Q}$, $\exists m, n \in \mathbb{Z}$ with $n \neq 0$ such that $\sqrt{p} = m/n$.

the Relation tab in the Symbols pane and locate that symbol, which can be inserted at present cursor location in edit pane, or the command code displayed on hover. As earlier, you can type quote marks as we show above, or when you begin to type a quote then accept the `\<open>` suggestion that pops up on tooltip (and then the matching `\<close>` when you type the closing quotation mark).

Follow that line with another line, and `"coprime m n"`²⁴ to obtain 13:

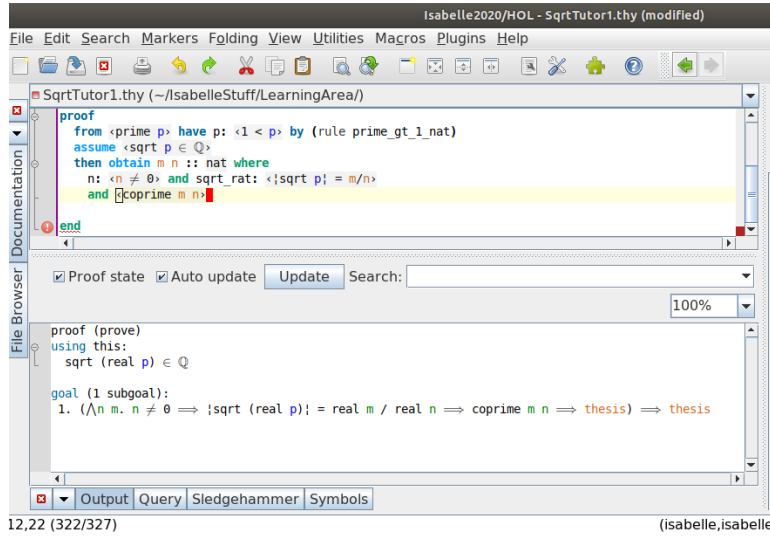


Figure 13. Subgoal proposed to obtain some facts from assumption $\sqrt{p} \in \mathbb{Q}$.

Looking at the Output pane at the bottom of the screen, we see our input has set up a new subgoal, i.e., to establish that our assumption $\sqrt{p} \in \mathbb{Q}$ allows us to infer that the absolute value of \sqrt{p} can therefore be represented by a ratio of two non-zero natural numbers (or positive integers) m/n where $\gcd(m, n) = 1$, i.e., (m, n) are coprime. Let us return to the Query pane (lower pane, click on Query tab) and see if Isabelle might give us a suggestion from selecting the Find Theorems tab, typing `solves` in the Find entry field and clicking on Apply (as we did earlier). We obtain the suggestion 14:

We see in the screen shot 14 that Isabelle believes (see the Query pane) the rule

Rats_abs_nat_div_natE

is a good schematic fit for our current subgoal, noting that the $?x$ schematic variables in the rule match the positions our \sqrt{p} proposed rational value holds in our subgoal statement previously viewed in the Output pane 13. It

²⁴From proof 2.1 our assumption that $\sqrt{p} \in \mathbb{Q}$ allows us to claim $\sqrt{p} = m/n$ with $\gcd(m, n) = 1$.

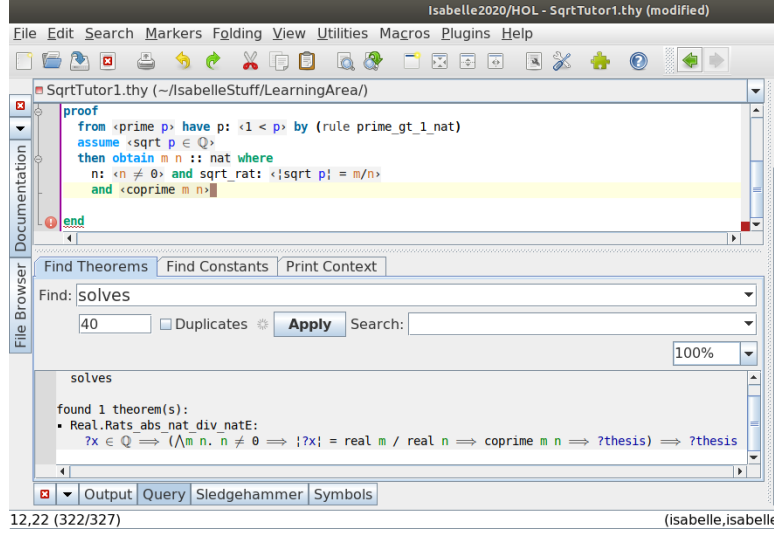


Figure 14. solves suggests a rule.

is also reassuring that the name of the suggested rule appears to describe what we are trying to establish, i.e., that a rational number can be represented by the ratio of two natural numbers, i.e., we are not blindly accepting inscrutable code, but rather output intended to allow a human to follow and approve the progress. Let us try using the previous method of applying a suggested rule, i.e., type `by (rule Rats_abs_nat_div_natE)` at the end of our current line in the edit area.²⁵

Clicking on the Output tab at bottom of jEdit screen, we see in the screen shot 15 that our new facts have been accepted (predicated on the assumption that $\sqrt{p} \in \mathbb{Q}$). If you do not obtain the indentation shown, use the Tab key on your keyboard to indent the relevant lines (we had to do that ourselves, probably because of the length of the last inputs and breaking over several lines).²⁶

If you click on the Query tab at the bottom of the jEdit window and select Print Context with context, terms, and theorems checked (click Apply to refresh display in that pane if necessary), you can obtain a detailed look at the status of the proof thus far Figure 16. In the screen shot you can see that we clicked and dragged the bottom of the edit area pane up (so the

²⁵If you are uncertain whether you have entered the same lines in the theory file at any point, you can always compare your edit area content with the raw print of the entire theory file in the Appendix A.

²⁶The logical block structure is known by Isar from keywords and syntax (implicit nesting of subproofs), but it is helpful to maintain a visual cue. This can also be accomplished by the use of `{...}` segments. See the discussion of proof context §2.2.1 in the *Isar Reference* [24].

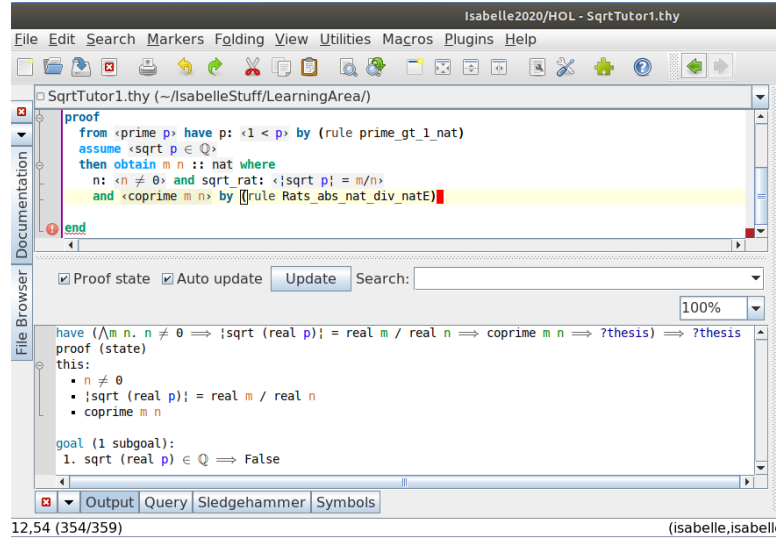


Figure 15. Rule application solves subgoal.

edit area is mostly hidden) in order to display all of the lines in the Query Print Content pane.

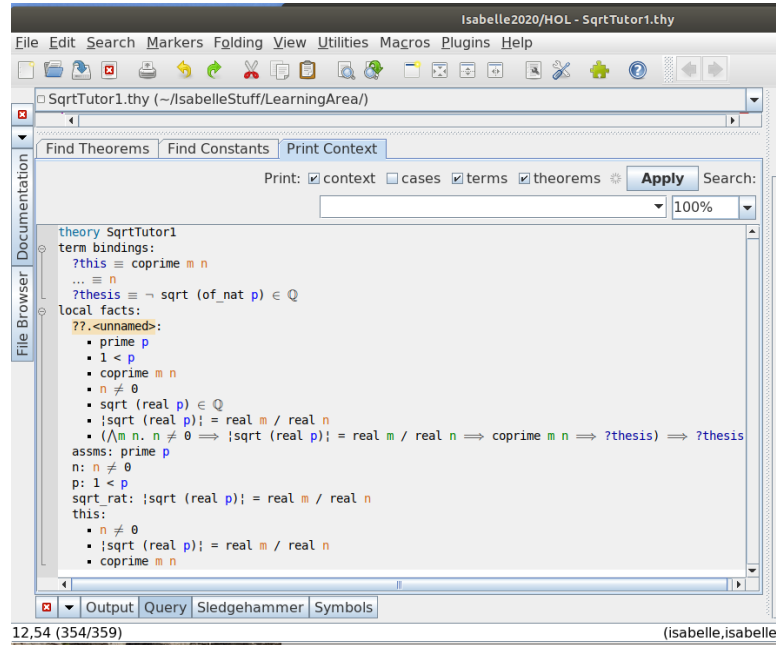


Figure 16. Context of the proof at line 12, Query pane.

In that Context display you may see that the local facts, that is the proof context within this subproof nesting, includes the assumption $\sqrt{p} \in \mathbb{Q}$ that we made to begin this sub-block (and that the outer level term binding `?thesis` is still to eventually prove $\neg P(p) = \sqrt{p} \in \mathbb{Q}$). We find it useful to know where jEdit provides information on the current logical context.

You may be puzzled as to why the Context labels p, m, n as `real` (we were). If you type the command `value p` (on a new line in the edit pane), the Output pane responds with `"p" :: "nat"`. Repeating that command for `m` and `n` also tell us that these terms are understood to be $\in \mathbb{N}$. That is reassuring. Let us look further.

Go to the Query tab at the bottom of the jEdit window, i.e., change the lower pane view from Output to Query. Click on the Find Theorems then type in `name: Rats_abs_nat_div_natE` in the Find field and click the Apply button. Among other things, we are told that a theorem was found containing that rule: `Real.Rats_abs_nat_div_natE`. That means that the `Real.thy` contains the rule `Rats_abs_nat_div_natE`.

Where is `Real.thy`? It is not strictly necessary for you to participate here, but if you feel comfortable navigating your computer and want to learn more about tracking down rules in theories, click on Utilities along the top of the jEdit window and look at the jEdit Home Directory. Remember the portion that ends at `src`. Alternately, you could click on the File Browser tab at the left-hand side of the jEdit window and look for the Path to Isabelle2020 (depending on how much directory navigation you have done it may already be pointing to HOL, which is where we want to look for `Real.thy`).

Now go to the top of the jEdit window and click on Search \Rightarrow Search in Directory. Copy `Rats_abs_nat_div_natE` to the Search for pane in the Search and Replace window that popped up. At the bottom of that new window, type in the Filter field, `Real.thy`. Assure that the Directory field points to the location of the Isabelle HOL src, on a Linux machine that will be something like `/usr/local/Isabelle2020/src/HOL/`. You should see something like the following screen shot before clicking the Find button Figure 17.

Now hit the Find button and see Figure 18

Click on the lemma line in the HyperSearch Results (not sure that is necessary, but cannot hurt, particularly in a case where there are multiple results). Close both of those search windows and click on the left-hand tab File Browser to close that folder side pane if it is open. You should now find jEdit has opened the `Real.thy` and taken the view in the edit pane to the beginning of the `Rats_abs_nat_div_natE` lemma Figure 19.

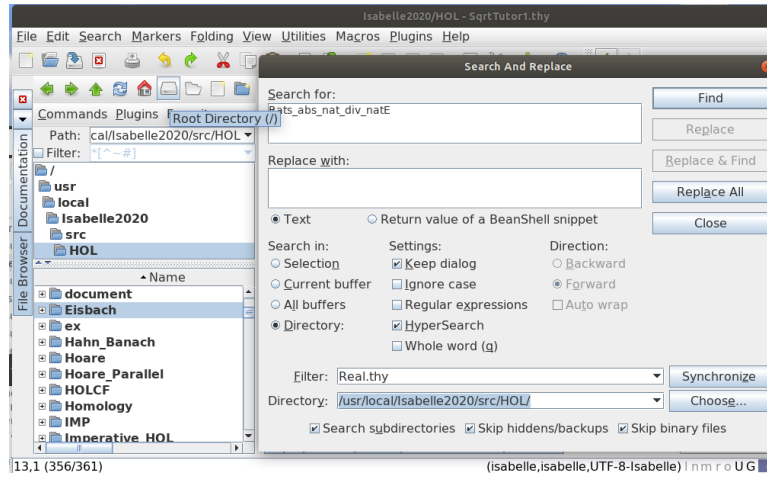


Figure 17. Searching for Real.thy before hit Find button.

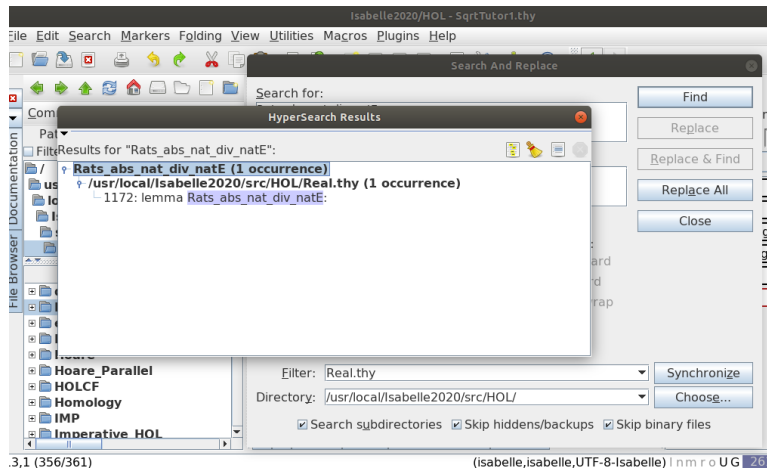


Figure 18. Searching for Real.thy before after hit Find.

Notice that all of the lines in the edit area view of the lemma are light red highlighted and the Output pane below the edit area is blank (click on the Output pane tab if not currently active). Both of these visual cues tell you that we are simply viewing this theory file rather than loading it to execute and check. This is because it is part of the Isabelle/HOL heap image that is loaded automatically when you start Isabelle. If we actually wanted to execute each line we would have to tell Isabelle to load an image

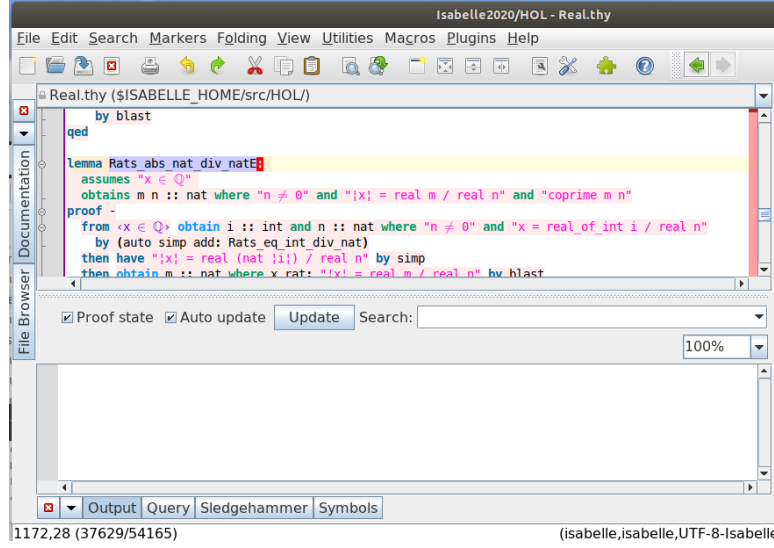


Figure 19. View the lemma in the Real.thy.

that does not already include this theory. We will not discuss this further here because it is quite a digression and not of immediate relevance.²⁷

Looking at `Real.thy` in the edit area, we see that its goal (the `obtains` line) is indeed the result we sought, i.e., the fact that a quantity is in \mathbb{Q} permits a representation as a ratio of two natural numbers m, n , coprime and $n \neq 0$. Scrolling down to examine the proof, we find it is set up with integer numerator and natural non-zero denominator, both coerced to real to obtain the division operation. That is established by a rule `Rats_eq_int_div_nat`, rational numbers equal integers divided by natural apparently. To establish that holds for the absolute value of the rational variable we again see the real coercion in the division operation. What we want to chase down is what is happening here with the use of the token `real` in an apparent type coercion role in these ratio division operations involving natural and integer numbers.

Accordingly, we will tell you how to locate the definition of the `real` term in this `Real.thy`²⁸ Select Search \Rightarrow Find on the top menu items of the jEdit window. When the Find subwindow appears, type in `quotient_type` in the Search for field, assure radio button Text is selected, along with

²⁷You can select a new logic to load on startup via the Plugins \Rightarrow Plugin Options \Rightarrow Isabelle \Rightarrow General. About four rows down from top of that window is a drop down menu showing HOL. If you click on the right drop down arrow it displays a very long list of alternate logics. Do not change this unless you have researched the subject and understand what you are doing.

²⁸We made a more efficient search of this theory file elsewhere so know it is a `quotient_type`.

Current buffer and direction Backward, and hit the Find button, then seeing something like the following Figure 20.

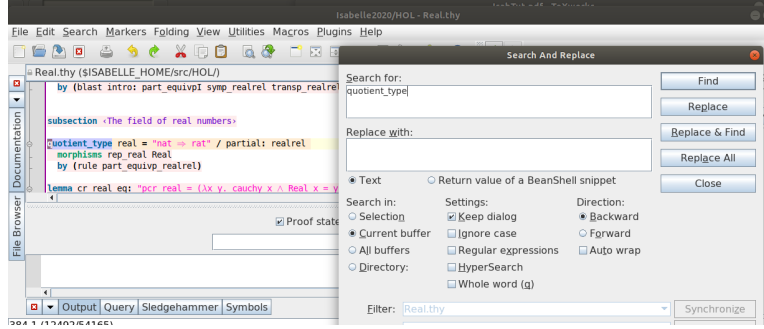


Figure 20. Search Real.thy for quotient type.

You may close the Search and Replace window and inspect the edit area now showing the `quotient_type` declaration for `real` in the `Real.thy`. We may be digressing excessively by this point, so we will summarize what we find here (and in the references we will cite).

Isabelle/HOL organizes its numeric theories using axiomatic type classes, e.g., *ring* and *field* with the usual abstract algebraic properties. The natural number type *nat* is a linearly ordered semiring, type *int* is an ordered ring, and type *real* is an ordered *field* (see §8.4.5 in [29]). Properties may not hold for a particular class, e.g., no abstract properties involving subtraction hold for type *nat* (since, of course, one might end up with a negative number, which would not be a natural number). Instead specific theorems are provided addressing subtraction on the type *nat*. More to the point, “all abstract properties involving division require a field.” [29]

What we are seeing here is a `quotient_type` being used to *lift* a division of natural or integer types to the abstract real type in order to satisfy the field requirement (see §11.9 of [24]). The quotient type *real* is created from the equivalence relation (definition `realrel` in the `Real.thy` file) between the underlying vanishing differences of Cauchy sequences [30]. We conclude that the appearance of `real` coercion terms in the ratios of our natural numbers (or positive integers) in our Isabelle proof states is appropriate and in fact required to satisfy rigor.²⁹

²⁹We should note, though, that a researcher in the proof automation field, Mathias Fleury, explained that the `sqrt` function in this code is only defined over the real numbers, hence the coercion by Isabelle. After that the ratio m/n was naturally coerced also. He advised that this behavior could be avoided by not overloading operators, e.g., not using a `sqrt` that applied to natural and integer numbers with coercion to real, but rather a dedicated function designed only for the target type. For example, a group formalizing some problems from the International Mathematical Olympiad [35] avoided using real numbers in a number theory proof by defining

Let us again return to our work on the `SqrtTutor1.thy` theory. Go to jEdit File → Close global (*not* Close All global) to close `Real.thy` and delete it from the jEdit buffers, at which time our `SqrtTutor1.thy` will reappear in the edit area. Then click on the left hand File Browser tab to close that pane. Type in the edit area a new line³⁰:

from n and sqrt_rat have "m = \<bar> sqrt p \<bar> * n "

Notice that in the Output pane below the edit pane we obtain two facts identified by the labels `n` and `sqrt_rat` we used earlier and that our new subgoal is stated Figure 21.

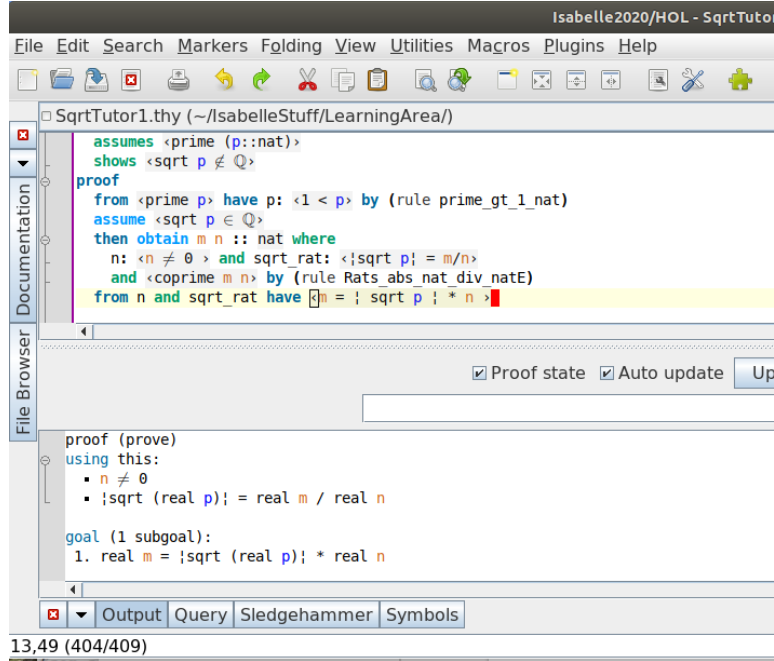


Figure 21. New proof line.

This appears to be a good time to let the simplifier `simp` just rewrite the line we gave it using the two labeled facts, so finish the line with `by simp`. The Output pane immediately updates our fact state to include this new line, i.e., it is proved. The Simplifier uses rules it knows about from the background theory or local proof context (our case here), along with chained facts

a natural number only square root (we would have had to redesign our entire proof to enjoy its benefits). We also found an Isabelle implementation of the $p \text{ prime} \rightarrow \sqrt{p} \notin \mathbb{Q}$ proof in ZF (Zermelo-Fraenkel Set Theory [19]) that used no real number references at all, but it is a difficult read (because of the subject complexity, as well as the fact that it was written prior to the Isar proof language, so the proofs are more machine scripts).

³⁰From our proof 2.1, since $\sqrt{p} = m/n$, we may obtain $m = |\sqrt{p}|n$.

and local assumptions (subgoal premises) to rewrite a statement, basically by substituting terms and expressions it recognizes (see §9.3 [24]).

Type in the edit area a new line following our work above³¹:

then have "m^2 = (sqrt p)^2 * n^2"

The Output pane updates to Figure 22.

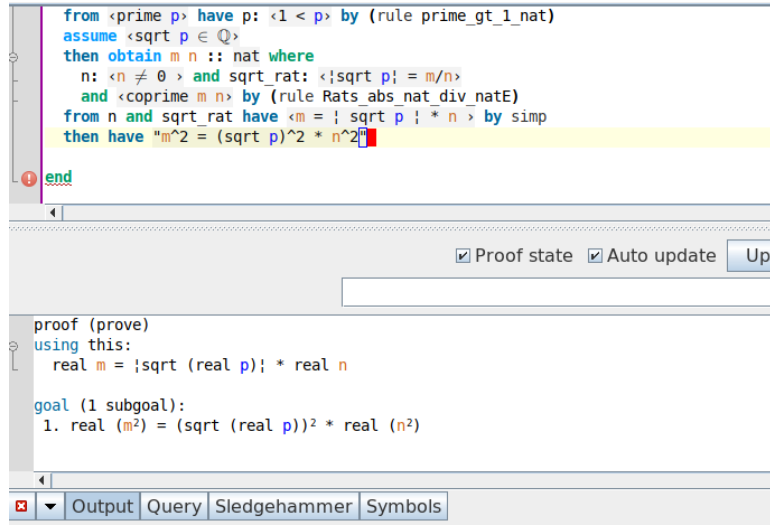


Figure 22. New proof line.

Let us switch to the Query pane (lower pane, click on Query tab) and see if Isabelle might give us a suggestion from selecting the Find Theorems tab, typing **solves** in the Find entry field and clicking on Apply (as we have done earlier). **solves** finds nothing, but in fairness, this is somewhat more complicated in that we want to recognize that squaring both sides of our last result are also equal. We can bring in the big guns for this one and try **Sledgehammer**.³² Click on the Sledgehammer tab next to the Query

³¹Following our proof 2.1, from $m = |\sqrt{p}|n$ it follows that $m^2 = (\sqrt{p})^2 n^2$.

³²Sledgehammer heuristically selects a few hundred of relevant lemmas from the currently loaded libraries. The default filter for that selection process is a combination of a relevance filter and a machine learning application which combines naive Bayes with k -nearest neighbor filtering. Sledgehammer then passes those facts along with the current theory proof goal to available automatic theorem provers (ATPs) and satisfiability-modulo-theories (SMT) solvers. If a proof is returned from the provers within a timeout limit, Sledgehammer tries to construct a proof using the builtin tactics of Isabelle/HOL. These will usually be short or one-line tactics that you can insert in your proof. You can also request Isar proofs. [32] See also §4.2 in [31] A *tactic* is a function that maps a goal state, i.e., a theorem, to a sequence of potential successor states. For example, an *Elim-resolution* tactic resolves the target goal with a rule, proves its first premise by assumption and removes that assumption from any new subgoals. *Forward-resolution* leaves the selected assumption in place. In general, assumption tactics close a subgoal by unifying some of its premises against its conclusion. See §4.2.1 in *The Isar Implementation* [26].

tab at the bottom of the window. In the Provers pane you should see the default provers that Isabelle includes (and is configured to use³³). We have run Sledgehammer throughout this tutorial with the “Try methods” box checked, so you should assure that is checked also (if you want the best chance of obtaining similar results as we show^{34, 35}). Click on Apply to set them loose on our current subgoal. After a brief wait while your computer cores are exercised, you obtain a list of results by the various provers (**z3** timed out, i.e., could not find a suggestion³⁶): Figure 23.

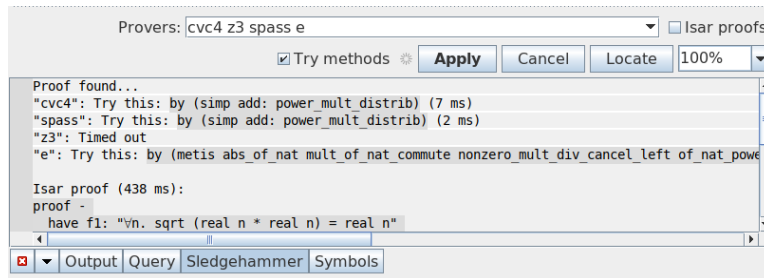


Figure 23. Sledgehammer prover results.

To try one you only have to click on it in the window to insert it automatically in the edit window at the current cursor location (the active part of the line is dark shaded and will highlight when you hover cursor there, a click at that point will cause the insertion).³⁷

We click on the first choice:

³³The **vampire** first-order resolution prover is included, as of this writing, with the Isabelle distribution and is located in the `Isabelle2020/contrib` folder. That path location should already be listed in the `Isabelle2020/Isabelle/etc/components` text file (as a local folder to the installation, i.e., not the entire path). We could not persuade Isabelle to use **vampire** until we made a new setting in `Isabelle/Jedit Options` saying to use Vampire noncommercial (changed unknown to yes) and typed in **vampire** on the prover list with the other provers in the Sledgehammer pane. We omitted it in this tutorial to avoid distracting the reader.

³⁴However, the results are not exactly determined, e.g., we obtained a slightly different set of suggestions from Sledgehammer on another occasion, possibly due to having loaded the entire theory file first. This should not worry you, since Isabelle explicitly replays all proof tool results through the LCF kernel before accepting them (we discussed the ML core briefly in our Introduction 1), i.e., a tool could fail or diverge unexpectedly, but is not allowed to return wrong results (see Chapter 5 in [27]).

³⁵The popup note if you hover on “Try methods” says it will try methods like **blast** and **auto** as *alternatives* to the **metis** method if checked, however you may see **metis** results, presumably if the particular prover did not succeed with the standard methods. Notice that we did not check the “Isar proofs” box either (but we do select that farther below in order to demonstrate that capability).

³⁶The **z3** SMT solver was developed Microsoft Research. It frequently times out without providing a suggestion in the current context, so we usually do not include it on the provers line.

³⁷§3.9 of the *jEdit* reference manual [23] describes this behavior as “proposed proof snippets are marked-up as *sendback*, which means a single mouse click inserts the text into a suitable place of the original source.”

"cvc4": Try this: by (simp add: power_mult_distrib)

Return to the Output pane to see whether that suggestion proved the current subgoal: Figure 24 (we see immediately that the suggestion was inserted in the edit pane).

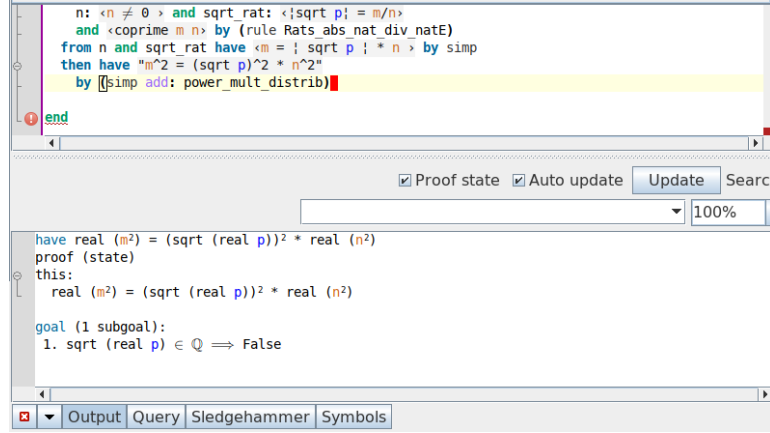


Figure 24. Sledgehammer suggestion proved current subgoal.

We see the subgoal has been proven (Output pane **this**). We will not go through it, but you may have noticed in the Sledgehammer pane that a multistatement Isar proof was also suggested (many or all of the suggestions returned might work, but we continue without investigating that possibility). This might be instructive to inspect (but we will not do that here), though we were pleased to require only the single suggestion of `cvc4`.³⁸

Let us continue with the proof. Enter the following new line in the edit area³⁹:

also have "(sqrt p)^2 = p"

This seems like another case where the Simplifier would easily prove the statement, so type `by simp` at the end of the line (or next line; the placement is not critical). The Output pane updates to indicate **this** is now the proved statement (we will skip the screen shot, assuming you are becoming acclimated to this environment). On the next line type:

also have "... * n^2 = p * n^2"

³⁸`cvc4` is an SMT solver, i.e., a Satisfiability Modulo Theories solver. See the *Sledgehammer* documentation that accompanies the Isabelle distribution for more information about the source of these provers. [32]

³⁹We are working towards $m^2 = pn^2$ in our proof 2.1, but every detail must be proven.

The “...” tells Isabelle to fill in the right-hand side of the previously proven statement.⁴⁰ If you look at the Output pane at this point, you will see the resulting subgoal has indeed substituted p for the “...” term 25 (this merits a screen shot and you can also check your current status in the edit pane).

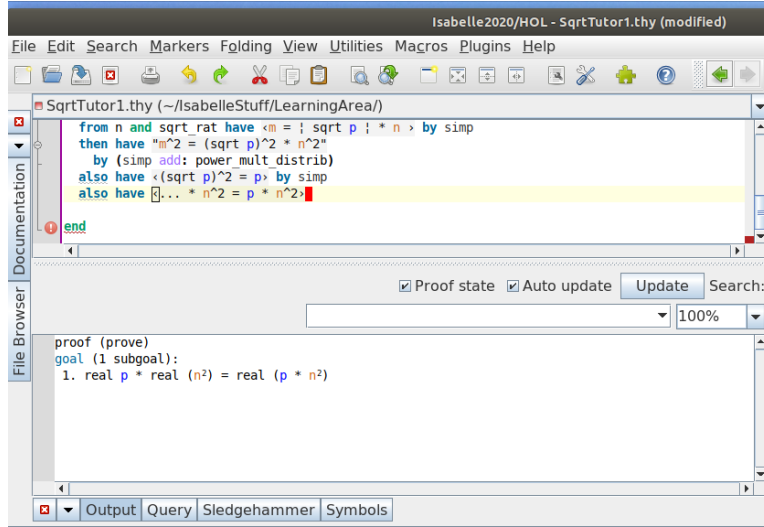


Figure 25. In Output pane the ... term is replaced by the previous result rhs.

That appears to be easily proven, so type `by simp` at the end of the line (or next line; the placement is not critical). The Output pane updates to indicate **this** is now the proved statement. On the next line type:

`finally have eq: "m^2 = p * n^2"`

Is this an easy subgoal? If we type `by simp` at the end of the line, the Output pane gives us an annoyed red failure message and the `by simp` command in the edit pane has also been colored to indicate displeasure. Erase `by simp` and let us invoke Sledgehammer. Select the Sledgehammer tab in the lower pane, erase the `z3`⁴¹ prover in the list (retain only `cvc4 spass e`) and hit Apply. All three provers quickly arrived at the suggestion `using of_nat_eq_iff by blast`, so mouse click on one of those lines in the Sledgehammer pane to insert that phrase into the current cursor location in the edit pane. Return to the Output pane to see the subgoal is proven Figure 26 .

⁴⁰To be precise, the term ... refers to “the argument of the last explicitly stated result (for infix notation this is the right-hand side).” See §1.2.1 of the Isar Reference [24].

⁴¹The `z3` prover timed out on our test and also disabled our system monitor for minutes, so we thought we would spare you the discomfort.

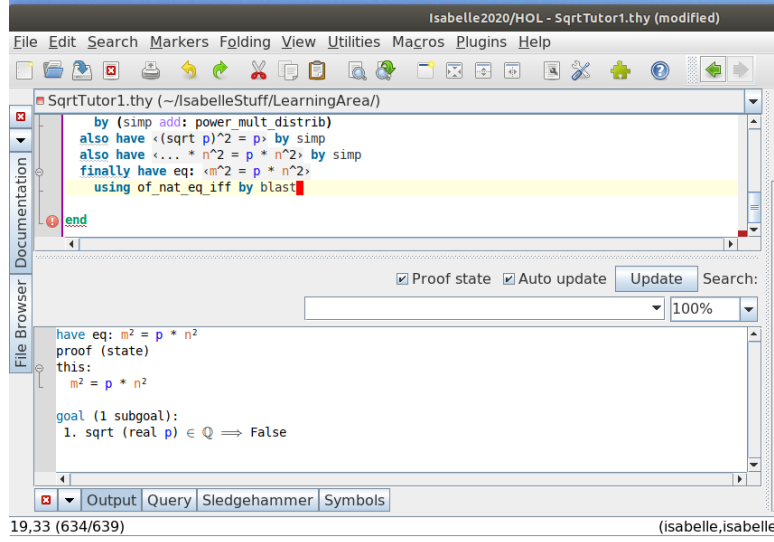


Figure 26. Blast the subgoal into submission.

We note, from §9.4.4 of the *Isar Reference* [24], that **blast** is a (separate) classical tableau⁴² prover that uses classical rule declarations (see §9.4.4 of the *Isar Reference* [24] for more information about when to use **blast**).

Let us type in the next line of the proof⁴³:

then have "p dvd m^2"

If you select the Query tab on the lower pane and then select the Print Context tab above the pane, you can clear the checkboxes and then check only the **term** box. When you do that you see the current term bindings, i.e., the last proven fact (**?this**) and the current goal (**?thesis**) Figure 27 .

We could get away with **by simp** on this relatively simple subgoal, but let us try the default refinement step, **by standard**, which is abbreviated with simply **..**. Typing that **..** at the end of our current line we see (return lower pane to Output tab) Isabelle immediately uses the fact that $m^2 = pn^2$ to arrive at $p \mid m^2$, i.e., the equivalent of dividing both sides of the known equation by p Figure 28 .

Type the next proof line:

⁴²A tableau is a graph-based representation of a set of sequents.[20] Sequent calculus is a generalization of natural deduction that is easier to automate. A sequent has the form $\Gamma \vdash \Delta$, where Γ and Δ are sets of formulae. If P_1, \dots, P_m represent assumptions and Q_1, \dots, Q_n represent alternate goals, then the sequent $P_1, \dots, P_m \vdash Q_1, \dots, Q_n$ is valid if $P_1 \wedge \dots \wedge P_m$ implies $Q_1 \vee \dots \vee Q_n$ (see §9.4.1 in [24]).

⁴³From our proof 2.1, since $m^2 = pn^2$, we may say that $p \mid m^2$ and also that $p \mid m$, since a prime divisor of a product of integers must divide one of the integer factors, which are identical in the case of a square.

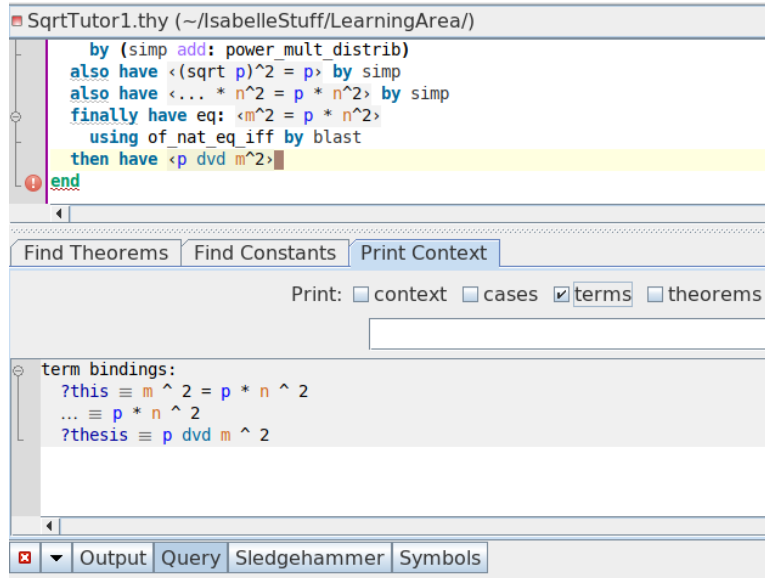


Figure 27. Relatively simple fact to current goal suggests standard method.

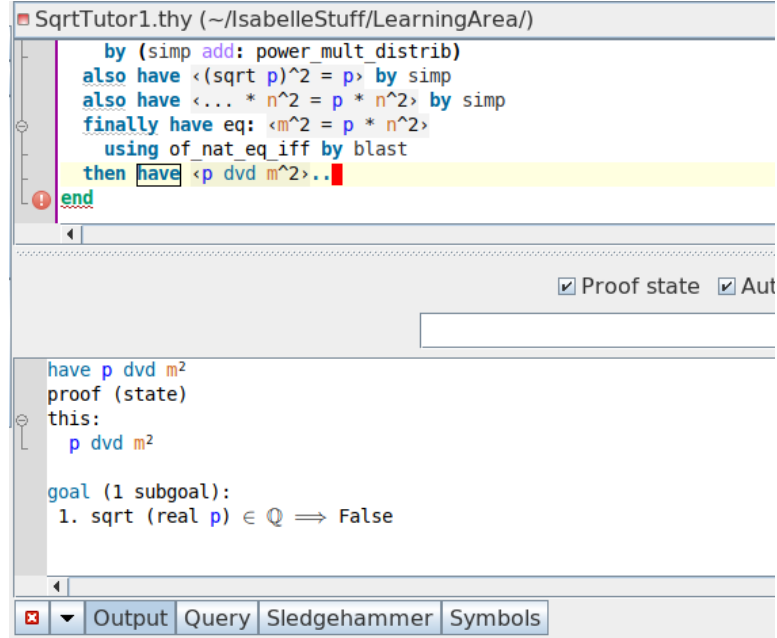


Figure 28. Default refinement step with by standard.

with "prime p" have dvd_m: "p dvd m"

We assume you recall that you may use the jEdit suggested open-close angle brackets instead of quotes. As we mentioned earlier, we show only quotes in this document, not having the exact font for the angle brackets. Looking at the Output pane after typing this line Figure 29, we see we require a fairly smart proof, i.e., a rule that knows that if a prime divides the power of a number, then it also divides the base number⁴⁴.

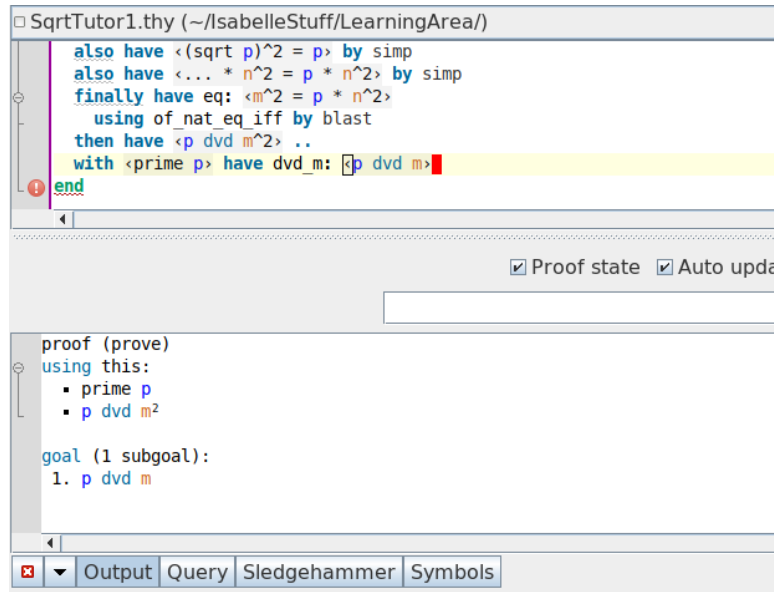


Figure 29. Want rule to prove prime divides base number of power.

A quick check of `by simp` does not prove our subgoal. Let us see what Sledgehammer might recommend. Select the lower pane tab for Sledgehammer and “loose the hounds”⁴⁵ with the Apply button. Very quickly Sledgehammer provides proof suggestions: Figure 30 .

We prefer the second suggestion (but all three should work), so click on the `cvc4` suggestion to use

```
using prime_dvd_power_nat by blast
```

Returning to the Output pane we see $p \mid m$ (`p dvd m`) has been proven. We enter the next line of the proof into the edit pane⁴⁶:

```
then obtain k where "m = p * k"
```

⁴⁴This could derive from the *prime divisor property*, i.e., a prime is something with the property that if it divides the product of two things, then it must divide one or the other (or both), paraphrasing Kimball Martin’s definition 2.2.2 in [34].

⁴⁵We kept `z3` off the list of provers for now, not wanting another timeout with odd incidental effects.

⁴⁶From our proof 2.1, because $p \mid m$ there must be an integer k such that $m = pk$.

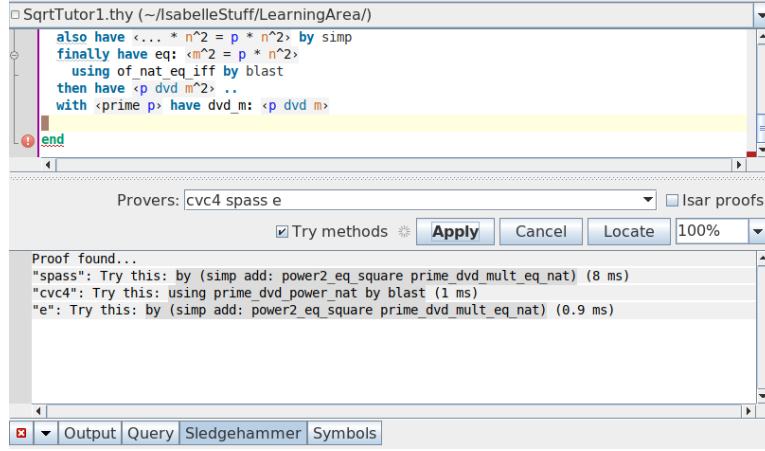


Figure 30. Sledgehammer results.

The resulting display of the new goal in the Output pane Figure 31 is interesting in that it shows some of the underlying logic framework, i.e., using universal quantification (a binder) to introduce a proof depending on a new term k . If there is such a k where $m = pk$, then we will add that fact to our local set of facts on the way to our goal.

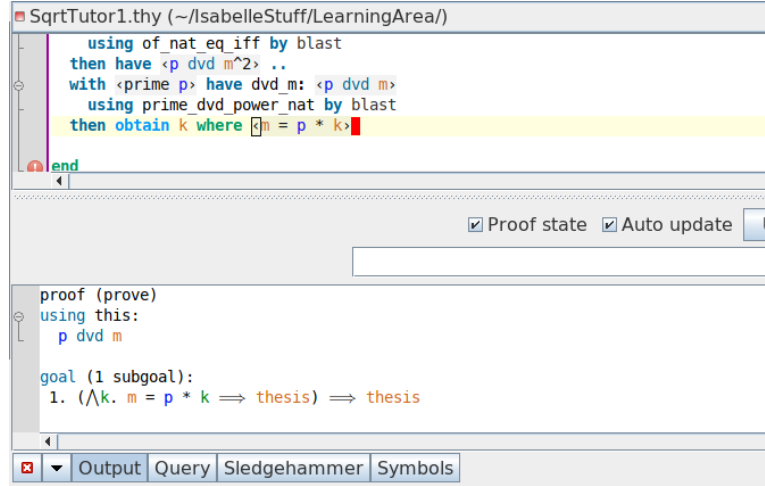


Figure 31. Quantifying a new term.

Let us make a quick try with the standard proof, i.e., type “.” at the end of this new line. That establishes the new fact $m = pk$. Enter our next line of proof in the edit pane⁴⁷:

⁴⁷From our proof 2.1, we already established that $m^2 = pn^2$. If we square $m = pk$ to obtain $m^2 = p^2k^2$, we can derive $pn^2 = p^2k^2$.

with eq have "p * n^2 = p^2 * k^2"

A look at the Output pane tells us that Isabelle is considering recent facts $m^2 = p n^2$ (because we referred to it with the label `eq` in the line we typed) and $m = p k$ (that being the most recent result). However, that does not make the connection we require obvious, i.e., that if we square that previous result $m = p k \rightarrow m^2 = p^2 k^2$ we could equate both expressions for m^2 . We will bring in the Sledgehammer again. Click on the Sledgehammer tab below the lower pane and hit Apply.

The Sledgehammer results on this occasion require more time than usual and two of the provers, `e` and `spass`, returned `metis`⁴⁸ method suggested solutions, with fairly complex parameters Figure 32.

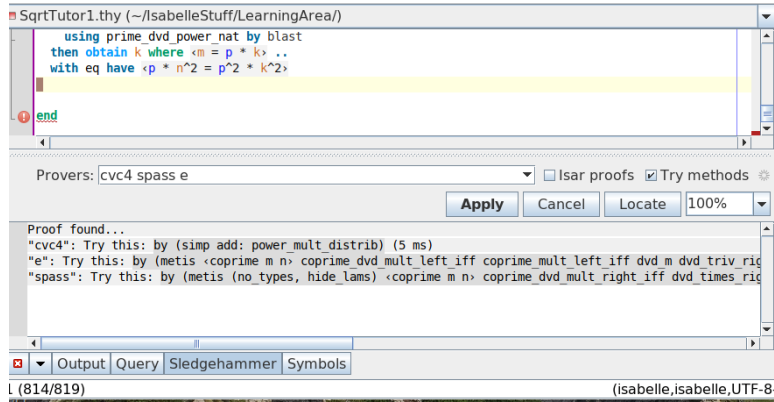


Figure 32. Some Sledgehammer suggestions including `metis` method.

This might be a good time to request Isar proofs on a Sledgehammer run in order to demonstrate clearly what that kind of result looks like. In the Sledgehammer pane, put a check in the “Isar proofs” checkbox to the right of the Provers list field. Then click on Apply again. Now we see both one-liners (proof suggestions using the `by` command and a method) and actual Isar proofs returned by the two ATPs `spass` and `e` Figure 33 (we clicked on the horizontal bar between the edit pane and the lower Output/Sledgehammer pane and dragged it up a bit so you can see the entire list of results returned;

⁴⁸From the *Isar Reference* [24] §12.6, `metis` is a model resolution theorem prover. From [32] we find ATPs or automatic theorem provers include `spass` and `e`, of the provers in our Prover list (also `vampire`, which as we mentioned earlier above, requires some slight configuration to run, though it is included with Isabelle). `cvc4` is categorized as an SMT solver, as is `z3`. The real utility of `metis` is in reconstructing (to run on the Isabelle kernel) proofs returned by provers: When given a list of facts that Sledgehammer found external provers (`spass`, `e`, `z3`, `vampire`, etc.) successfully used in a proof, `metis` re-finds the proof (if possible), reconstructing it so it can be validated on the Isabelle inference kernel, that being the essential to the LCF philosophy (see §3.4 in [33]). Those reconstructions are what `metis` is showing us in Figure 32.

you can also undock the pane, and any of the other panes, and float it separately, produce multiple instances, etc. but this is too much for us and probably too much for our limited computer memory).

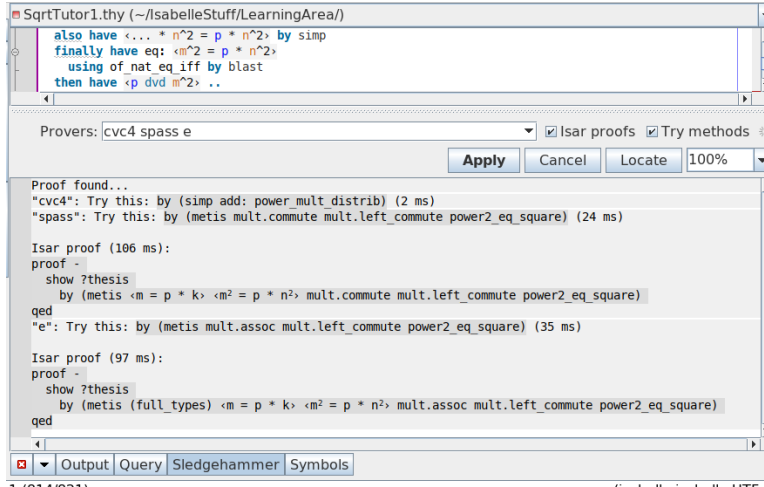


Figure 33. Sledgehammer returning Isar proof on request.

Go ahead and click on the shaded portion of the first Isar proof, the one directly beneath the **spass** one-line suggestion. As a result, that Isar proof block is inserted into the edit pane at the current location in the theory (return to Output pane to see result): Figure 34

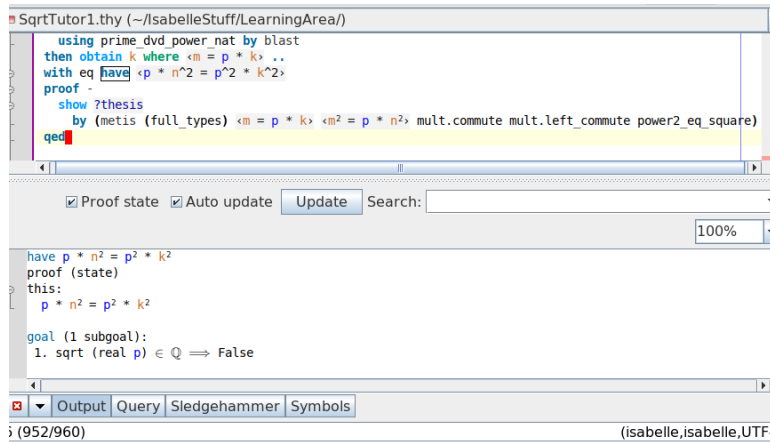


Figure 34. Isar proof inserted.

The desired goal, $p n^2 = p^2 k^2$, is proved (Output pane). Notice that we inserted an entire subproof block into our theory, complete with a beginning **proof**– command and ending **qed** to close the subblock and export

its result back to the enclosing level. That $-$ (a minus sign) following the `proof` command tells Isabelle not to execute the usual single reduction step (in which Isabelle tries to pick an appropriate rule automatically from the current context, as in the double-dot `..` step we have used earlier, see §6.4.3 of [24]). In this particular case we see that the Isar proof returned is indeed more informative of how the subgoal was proved, i.e., by commuting to the left pn^2 and squaring $m = pk$, then setting the left side equal to the newly squared right side since they both were equal to the same term, m^2 . We may as well keep this proof and continue.

Type in our next proof line into the edit pane:

```
with p have "n^2 = p * k^2"
```

Let us see another Isar proof if Sledgehammer is able. Return to the Sledgehammer pane and, leaving the “Isar proof” box checked along with “Try methods,” click on Apply. Well, Sledgehammer worked for a while on this one and one of the ATPs, `e`, failed to construct an Isar proof. `spass` was able to construct an Isar proof, something to the effect that since p was greater than zero, we could divide both sides of $pn^2 = p^2k^2$ by p (cancel left) and obtain $n^2 = pk^2$. Click on the shaded area of that Isar proof in the Sledgehammer pane to insert the block into our edit pane. Return to the Output pane to see that did prove our goal. $n^2 = pk^2$.

Type the following new proof line in the edit pane following our recent subproof:

```
then have "p dvd n^2"
```

This appears fairly simple, just noting that we could divide both sides of the last result by p , so let us try the double-dot standard proof by typing “`..`” at the end of that line. That proves this subgoal. Let us maintain our rapid pace⁴⁹ and enter the next line of the proof:

```
with "prime p" have "p dvd n"
```

This line is identical to a previous line, with n replacing m , namely

```
with "prime p" have dvd_m: "p dvd m"
```

We proved that previous line using `by (rule prime_dvd_power_nat)`. Let us try that again here, typing it in at the end of the current line. That proves the new line Figure 35

We have now proved, on the assumption that $\sqrt{p} \in \mathbb{Q}$ in our proof 2.1 above for Theorem 2.1, that:

⁴⁹If you are not sure you have typed in all theory lines, see the Appendix A where we give you the raw text lines of the entire theory file.

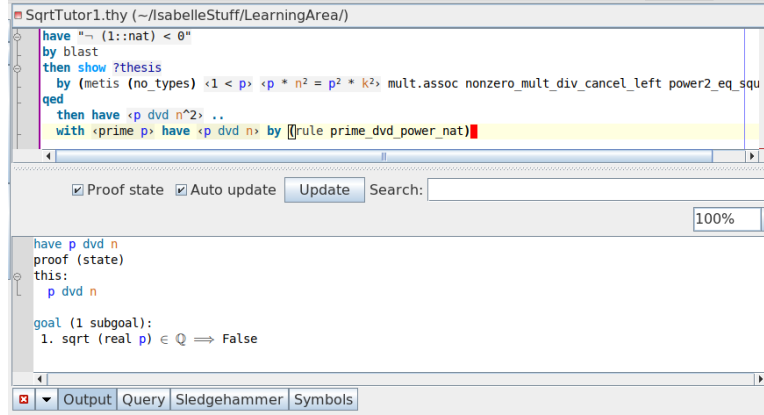


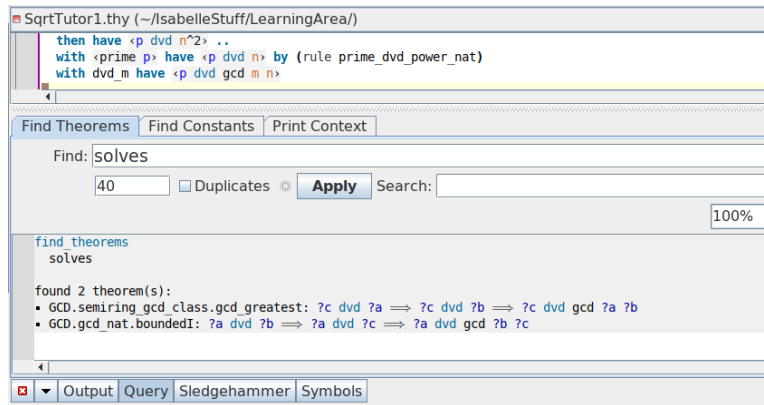
Figure 35. Reuse previous similar proof one-liner.

p prime divides both m and n and therefore must divide the greatest common divisor of m, n , i.e., $p | \gcd(m, n)$

Let us enter that into our proof, using the current fact that $p | n$ (see Output pane “this”) and the previous fact $p | m$ by referring to the label we have it above, `dvd_m`:

with `dvd_m` have “`p dvd gcd m n`”

There should be a rule that applies to this use of properties of the gcd. We can use the `solves` criterion in the Query pane Find Theorems as we did earlier. Switch from the Output to the Query tab on the lower pane. Then select the Find Theorems tab and type `solves` in the Find entry field and click on Apply. Isabelle finds two theorems that are closely related to our current goal. We prefer the first one, `gcd_greatest` Figure 36

Figure 36. Obtain `solves` suggested rule for gcd context.

Let us copy `gcd_greatest`⁵⁰ and construct a `by (rule)` application, typing that in at the end of our current line to obtain Figure 37.

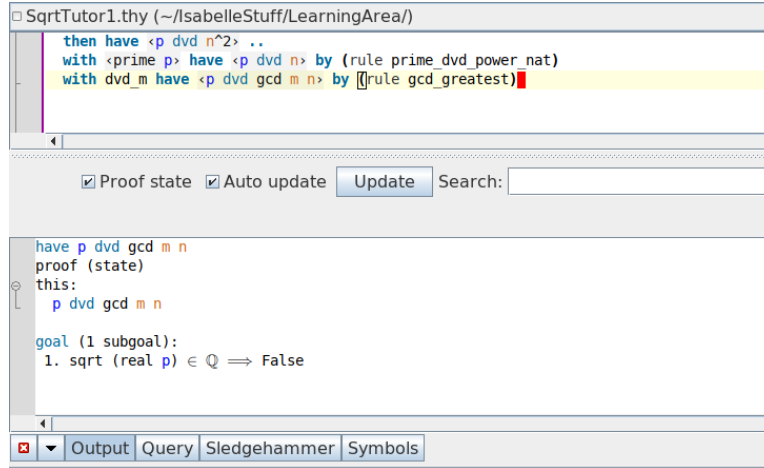


Figure 37. Apply `solves` suggested rule for `gcd` context.

That proves this new subgoal, but in our Theorem 2.1 “we stipulated that m, n are relatively prime, i.e., that $\gcd(m, n) = 1$.” Type in the following line in the edit pane to bring this observation into consideration:

```
with "coprime m n" have "p = 1"
```

This appears to be a relatively simply statement, so try the Simplifier, typing `by simp` at the end of that line. We then see that this $p = 1$ statement is proven. If you switch the Output pane to the Query pane, go to the Print Context tab, deselect all checkboxes then place a check in the `theorems` box, that pane should update to list all the established facts (`theorems`). Scroll down the long list to see the last three, i.e., Figure 38.

We have two contradictory facts, $1 < p$ and $p = 1$. We are ready to administer the *coup de grâce*. Return the lower pane to the Output and type into the edit pane

```
with p show False
```

The state is then shown to be `prove` using $1 < p$ and $p = 1$ with goal `False`. It seems a pity that it will be so easy now (but we did work to arrive here), but we need only type `by simp`. The Output pane updates to Figure 39.

The Output pane tells us the original goal to `show False` was successfully obtained when our local proof (we have been working on the assumption

⁵⁰We do not need to type in the entire fully qualified name:

`GCD.semiring_gcd_class.gcd_greatest`, just the portion to the right of the rightmost period.

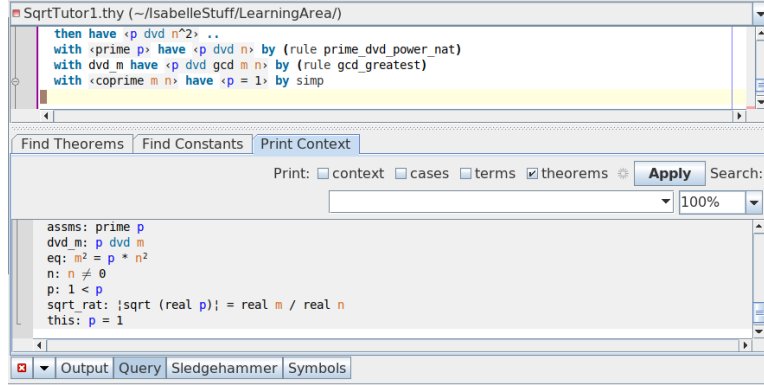
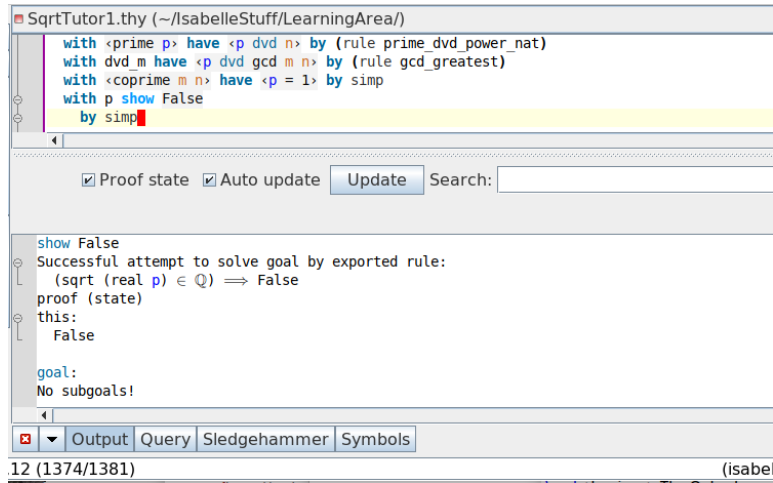


Figure 38. Facts now establish a contradiction.

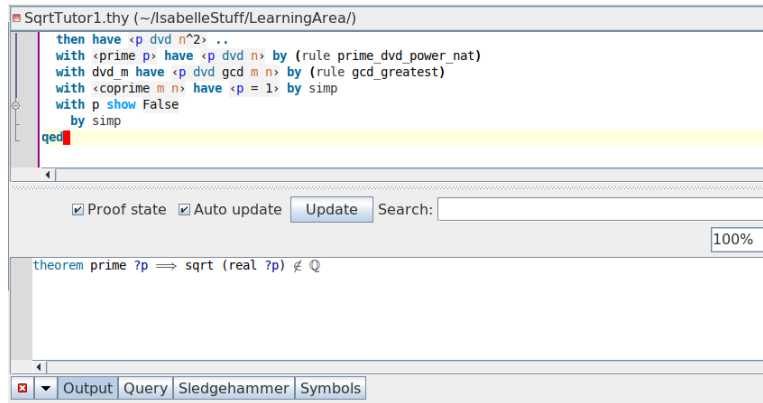
Figure 39. We proved $\sqrt{p} \in \mathbb{Q} \implies \text{False}$.

that $\sqrt{p} \in \mathbb{Q}$ in a local proof contained within the outer proof with original goal) exported the local result $\sqrt{p} \in \mathbb{Q} \implies \text{False}$ (because p could not be equal to 1 and greater than 1 simultaneously), closing that local context. We may terminate our proof now, typing `qed` Figure 40.

The Output pane now shows the theorem we have established, complete with schematic variable $?p$,⁵¹ which could be instantiated by use of the theorem elsewhere (if we named the `theorem` to provide a reference⁵²). Our proof of Theorem 2.1 with Isabelle/HOL is completed.

⁵¹See §3.3.7 in the *Isar Reference* [24].

⁵²`theorem` and `lemma` are the same kind of entity, i.e., definitions requiring formal justification by proof (refer to §5.1 [24]). Their different names serve as formal comment. If we gave our `theorem` a name, e.g., `prime_not_q`, we could invoke it.

**Figure 40.** The theorem is completed.

Closing Thoughts

We hope our tutorial has made using Isabelle as a proof assistant in the development or analysis of mathematical theorems seem a desirable possibility, particularly for the student¹. To assure we have not given a false sense of simplicity, we emphasize again that we were merely reproducing a theory already constructed and provided with the Isabelle distribution (`HOL/ex/Sqrt.thy`). The actual process of developing a new proof is considerably more challenging. See [22] (mentioned in our Introduction 1) for a somewhat dated tutorial for mathematicians featuring portions of new theory development in Isabelle, guided by existing relevant theories.

As far as obtaining guidance from existing theories relevant to your area of interest, the Isabelle distribution contains several different base logic systems in addition to the HOL (Higher-Order Logic) that we used above, e.g., you might do work in ZF (Zermelo-Fraenkel Set Theory) instead, or FOL (First Order Logic). Even within HOL alone there are many branches of mathematics implemented, e.g., there are sections including theories for Algebra, Analysis, Cardinals, Complex Analysis, Number Theory, and many more. The Isabelle distribution includes more than 700,000 lines of Isabelle/HOL theories[21]. If that were not enough, there is an online collection of proofs developed by the Isabelle community (obtain at [2]), more than 480 articles written by 322 authors, comprising 134,000 theorems and supporting lemmas in the areas of both computer science and mathematics (description from [21]).

¹Compared with the prototypes of the 1970s, Isabelle has already become a practical and versatile tool used by system designers, mathematicians and others, providing a highly readable structured proof language and interactive live proof development environment[21].

The entire `SqrtTutor1.thy` presented in Section 2.2 is given as raw text below. We inserted a few hard line breaks to prevent long lines running off the page margin on the right:

```
theory SqrtTutor1
  imports Complex_Main "HOL-Computational_Algebra.Primes"
begin
theorem
  assumes \ $p :: \text{nat}$ 
  shows \ $\sqrt{p} \notin \mathbb{Q}$ 
proof
  from \ $p$  have  $p > 0$ 
    by (rule prime_gt_1_nat)
  assume \ $\sqrt{p} \in \mathbb{Q}$ 
  then obtain  $m\ n :: \text{nat}$  where
     $n > 0$  and
     $\sqrt{p} = m/n$ 
    and  $\text{coprime } m\ n$  by (rule Rats_abs_nat_div_natE)
  from  $n$  and  $\sqrt{p}$  have  $m = \sqrt{p} \cdot n$ 
    by simp
  then have " $m^2 = (\sqrt{p})^2 \cdot n^2$ "
    by (simp add: power_mult_distrib)
  also have  $(\sqrt{p})^2 = p$  by simp
  also have  $\dots \cdot n^2 = p \cdot n^2$  by simp
  finally have  $m^2 = p \cdot n^2$ 
    using of_nat_eq_iff by blast
  then have  $p \mid m^2$  ..
  with \ $p$  have  $p \mid m$ 
    using prime_dvd_power_nat by blast
```

```

    then obtain k where \<open>m = p * k\<close> ..
    with eq have \<open>p * n^2 = p^2 * k^2\<close>
    proof -
      show ?thesis
        by (metis (full_types) \<open>m = p * k\<close>
          \<open>m\<sup>2 = p * n\<sup>2\<close>
          mult.commute mult.left_commute power2_eq_square)
      qed
    with p have \<open>n^2 = p * k^2\<close>
    proof -
      have "\<not> (1::nat) < 0"
      by blast
    then show ?thesis
      by (metis (no_types) \<open>1 < p\<close>
        \<open>p * n\<sup>2 = p\<sup>2 * k\<sup>2\<close>
        mult.assoc nonzero_mult_div_cancel_left power2_eq_square)
    qed
    then have \<open>p dvd n^2\<close> ..
    with \<open>prime p\<close> have \<open>p dvd n\<close>
      by (rule prime_dvd_power_nat)
    with dvd_m have \<open>p dvd gcd m n\<close> by (rule gcd_greatest)
    with \<open>coprime m n\<close> have \<open>p = 1\<close> by simp
    with p show False
      by simp
    qed
  end

```

Bibliography

- [1] *Isabelle Home, Technical University of Munich*, <https://isabelle.in.tum.de/>. Isabelle automated theorem prover and interactive proof assistant available for download here. Developed at University of Cambridge and Technical University of Munich.
- [2] *Archive of Formal Proofs*, <https://www.isa-afp.org>. Online collection of proofs contributed by the Isabelle community.
- [3] Jørgen Villadsen, Andreas Halkjær From, and Anders Schlichtkrull, *Natural Deduction and the Isabelle Proof Assistant*, EPTCS **267** (2018), 140–155, available at <http://dx.doi.org/10.4204/EPTCS.267.9>. Quaresma, P. and Neuper, W. Eds.: 6th International Workshop on Theorem proving components for Educational software.
- [4] Makarius Wenzel, *Interaction with Formal Mathematical Documents in Isabelle/PIDE* (May 5, 2019), available at <http://arxiv.org/abs/1905.01735v1>.
- [5] Roger Penrose, *The Road to Reality*, Jonathan Cape 2004, London, 2004. Chapt. 3.
- [6] INRIA, *Coq proof assistant* (2017), <https://coq.inria.fr/>.
- [7] Artem Yushkovskiy, *Comparison of Two Theorem Provers: Isabelle/HOL and Coq* (September 6, 2018), available at <http://arxiv.org/abs/1808.09701v2>.
- [8] Florian Haftmann, *The Isabelle/HOL type-class hierarchy* (April 15, 2020). This document accompanies the Isabelle distribution as type-classhierarchy.pdf.
- [9] ———, *Haskell-style type classes with Isabelle/Isar* (April 15, 2020). This document accompanies the Isabelle distribution as classes.pdf.

-
- [10] John P. Burgess, *Rigor and Structure* (2018), <https://www.princeton.edu/~jburgess>. 413 page essay by Princeton University philosophy professor.
 - [11] Sir Thomas Heath, *A History of Greek Mathematics* (1921), available at <http://www.archive.org/details/cu31924008704219>. Online text from a scan of the original Oxford published book at Cornell University Library.
 - [12] Jeremy John Gray, *Mathematics – Britannica Online Encyclopedia* (2020), available at <https://www.britannica.com/science/mathematics>. 2021 print from online article published by Encyclopaedia Britannica, Inc.
 - [13] Freek Wiedijk, *The Seventeen Provers of the World* (2004), available at <http://www.cs.ru.nl/~freek/comparison/comparison.pdf>.
 - [14] Mirna Džamonja, *Formalising Ordinal Partition Relations Using Isabelle/HOL* (November 30, 2020), available at <http://arxiv.org/abs/2011.13218v2>.
 - [15] Lawrence C. Paulson, *Isabelle’s Logics* (April 15, 2020). Includes contributions by Tobias Nipkow and Markus Wenzel. This document accompanies the Isabelle distribution as `logics.pdf`.
 - [16] ———, *Formalising Mathematics In Simple Type Theory* (April 20, 2018), available at <http://arxiv.org/abs/1804.07860v1>.
 - [17] ———, *The foundation of a generic theorem prover*, *Journal of Automated Reasoning* **5** (1989), 363–397.
 - [18] ———, *Isabelle: The Next 700 Theorem Provers*, *Logic and Computer Science*, 1990, pp. 361–386.
 - [19] ———, *Isabelle’s Logics: FOL and ZF* (April 15, 2020). Includes contributions by Tobias Nipkow and Markus Wenzel. This document accompanies the Isabelle distribution as filename `logics-FOL-ZF.pdf`.
 - [20] ———, *Logic and Proof* (2000). Course notes by author (lcp@cl.cam.ac.uk) for University of Cambridge Computer Science Tripos Part IB.
 - [21] Lawrence C. Paulson, Tobias Nipkow, and Makarius Wenzel, *From LCF to Isabelle/HOL* (July 17, 2019), available at <http://arxiv.org/abs/1907.02836v2>.
 - [22] B. Grechuk, *Isabelle Primer for Mathematicians* (2010), available at http://web.cs.wpi.edu/~dd/resources_isabelle/isabelle_primer_mathematicians.pdf. Technical report, University of Edinburgh, quick introduction to the Isabelle/HOL proof assistant, aimed

at mathematicians who would like to use it for the formalization of mathematical results.

- [23] Makarius Wenzel, *Isabelle/jEdit* (April 15, 2020). This document accompanies the Isabelle distribution as filename jedit.pdf.
- [24] ———, *The Isabelle/Isar Reference Manual* (April 15, 2020). This document accompanies the Isabelle distribution as filename isar-ref.pdf. Many others contributed to the document, too numerous to list here.
- [25] ———, *The Isabelle System Manual* (April 15, 2020). This document accompanies the Isabelle distribution as filename system.pdf.
- [26] ———, *The Isabelle/Isar Implementation* (April 15, 2020). This document accompanies the Isabelle distribution as filename implementation.pdf. Many others contributed to the document, too numerous to list here.
- [27] ———, *Interactive Theorem Provers from the perspective of Isabelle/Isar* (2014). Univ. Paris-Sud, Laboratoire LRI. author email: makarius@sketis.net.
- [28] Tobias Nipkow, *Programming and Proving in Isabelle/HOL* (April 15, 2020). This document accompanies the Isabelle distribution as filename prog-prove.pdf.
- [29] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel, *A Proof Assistant for Higher-Order Logic* (April 15, 2020). This document accompanies the Isabelle distribution as filename tutorial.pdf.
- [30] Sylvie Boldo, Catherine Lelay, and Guillaume Melquiond, *Formalization of Real Analysis: A Survey of Proof Assistants and Libraries* (February 18, 2014), available at <https://hal.inria.fr/hal-00806920v1>.
- [31] Mathias Fleury and Hans-Jörg Schurr, *Reconstructing veriT Proofs in Isabelle/HOL* (September 2, 2019), available at <https://hal.inria.fr/hal-02276530>.
- [32] Jasmin Blanchette, *Hammering Away, A User's Guide to Sledgehammer for Isabelle/HOL* (April 15, 2020). This document accompanies the Isabelle distribution as filename sledgehammer.pdf. Lawrence C. Paulson contributed to this document also.
- [33] Jasmin Christian Blanchette, Lukas Bulwahn, and Tobias Nipkow, *Automatic Proof and Disproof in Isabelle/HOL* (2011). Fakultät für Informatik, Technische Universität München.
- [34] Kimball Martin, *An (algebraic) introduction to Number Theory* (2017), available at <http://www.math.ou.edu/~kmartin/intro-nt/>. Author's lecture notes for upper-level undergraduate introduction to number theory for math majors at University of Oklahoma Fall 2017.

-
- [35] Filip Marić and Stojanović-Durdević, *Formalizing IMO Problems and Solutions in Isabelle/HOL* (2020). This is a paper by mathematics faculty at the University of Belgrade formalizing some problems from the International Mathematical Olympiad (IMO).

Index

- absolute value bars, **18**
- assumes command, **9**
- Auto update, **9**

- blast, classical tableaux prover, **31**
- block structure, proof, **7**
- browser preview, **8**

- cartouche, **13**
- chain state, **13**
- Close global, current buffer, **18, 26**
- Commensurable, **4**
- cvc4 prover, **29**

- division, abstraction with field, **25**
- drag resize panes, **20**

- executing viewed theories, **23**
- export local proof result to outer block, **40**
- export result to enclosing level, **36**

- facts, examine current, **16**
- File Browser, **16**
- File Save, **8**
- Find Theorem, solves, **16**
- Find Theorems, Query tab, **22**
- free variable, **10**
- from command, **13**

- heap image load, **23**
- HOL, **1**
- HyperSearch, jEdit, **22**

- imports command, **7**

- inference rules, theorem to theorem, **2**
- inner syntax, **9**
- insert rhs previous statement proved, ..., **30**
- insert Sledgehammer proof suggestion, **28**
- Isar, **1**
- Isar proof, inserted as sub-block, **36**
- Isar proof, Sledgehammer generated, **35, 37**
- Isar virtual machine, figure, **13**

- jEdit, **2**

- label for later reference, **14, 26**
- lemma, theorem same entity type, **40**
- lift numeric type for operation, **25**
- locate computer directory path, **22**
- locate rule in theory file, **22**
- logical block structure, **20**

- metis, **35**
- ML, **2**

- numeric type, examine with value command, **22**

- Object Logic, **24**
- Output pane, **8**
- overloading, **25**

- Panes, open or close, resize, **17**
- panes, panels, **6**
- PIDE, **2**

- Print Context, facts, assumptions, bindings, **22**
- Proof assistant, **1**
- proof results validated, **28**
- proof state, **11**
- Proof writing, **2**
- proof, completed with schematic variable, **40**
- prove by rule, **37**
- prove state, **14**
- prover list, **28**
- Punctuation tab, **18**
- Pythagoras, **4**
- Pythagorean theorem, **4**

- qed, terminate proof, **40**
- Query pane, Print Context, **14**
- quotation marks, **9**
- quotation marks, cartouche, **33**

- real type cast, **12**
- real type, quotients, **24**
- refer to labelled fact, or current, **38**
- Relation tab, **19**
- rule, copy into proof, **39**
- rule, proof suggestion, **15**

- schematic variable, **16**
- schematic variables, rule match, **19**
- Scratch.thy, **6**
- Search in Directory, jEdit, **22**
- session, **7**
- show False, use contradiction, **39**
- Sidekick, **11**
- Simplifier failure, **30**
- Simplifier, easy proof task, **29**
- Simplifier, rewrite with, **26**
- Sledgehammer, **27**
- Sledgehammer tab, **27**
- Sledgehammer, when simp fails, **33**
- solve_direct, **15**
- solves, Query pane Find Theorems, **38**
- Sqrt.thy, **5**, **42**
- square root function, **25**
- Square root of prime, **5**
- standard proof attempt, ..., **34**, **37**
- state mode, Isar, **12**
- subgoal, **14**, **19**, **26**
- suggestions, entry, **10**
- Symbols, select, **10**
- Syntax highlight, color, **9**

- tableaux, set of sequents, **31**

- term, binding, **31**
- term, check in Query Print Context pane, **31**
- Text area, main, **6**
- theorem command, **8**
- Theories side tab, **11**
- theory command, **8**
- ?thesis, **14**, **22**
- tool tips, **10**
- Try methods options, Sledgehammer, **28**
- type classes, **25**
- Type theory, simple, **1**
- type, variable, **10**

- universal quantifier, binder, **34**

- vampire, **28**

- with have, bring fact into consideration, **39**

- z3 prover, **28**
- Zermelo-Fraenkel Set Theory, **26**