Ordered Geometry in Hilbert's *Grundlagen der Geometrie*

Phil Scott

Doctor of Philosophy

Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh

2015

Abstract

The *Grundlagen der Geometrie* brought Euclid's ancient axioms up to the standards of modern logic, anticipating a completely mechanical verification of their theorems. There are five groups of axioms, each focused on a logical feature of Euclidean geometry. The first two groups give us *ordered geometry*, a highly limited setting where there is no talk of measure or angle. From these, we mechanically verify the Polygonal Jordan Curve Theorem, a result of much generality given the setting, and subtle enough to warrant a full verification.

Along the way, we describe and implement a general-purpose algebraic language for proof search, which we use to automate arguments from the first axiom group. We then follow Hilbert through the preliminary definitions and theorems that lead up to his statement of the Polygonal Jordan Curve Theorem. These, once formalised and verified, give us a final piece of automation. Suitably armed, we can then tackle the main theorem.

Acknowledgements

Many thanks go to my supervisor Jacques Fleuriot for his support and encouragement. Thanks to Laura Meikle for spotting a simplification of Theorem 3.11, and especially to Steven Obua for helping me prove transitivity of polygonal rotations described in §11.5.1.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except for work included which forms part of jointly-authored publications. Our contribution and that of the other authors to this work is explicitly indicated below. We confirm that appropriate credit has been given within the thesis where reference has been made to the work of others, and that this work has not been submitted for any other degree or professional qualification except as specified.

In Chapters 4 and 5, we expand on a combinator language which evolved between *Composable Discovery Engines for Interactive Theorem Proving* and *A Combinator Language for Theorem Discovery*, published respectively in *Interactive Theorem Proving* in 2011 and *Intelligent Computer Mathematics* in 2012. The intended application of this language was first described in *An Investigation of Hilbert's Implicit Reasoning through Proof Discovery in Idle-Time*, published in *Automated Deduction in Geometry* in 2010, and our analysis from this paper has been updated and can be found in §5.2.3 and §5.3.1.1. The work was co-authored with Jacques Fleuriot.

(*Phil Scott*)

Table of Contents

Chapter 1

Introduction

In this thesis, we recount our formalisation and mechanical verification of a subset of *synthetic geometry*. This style of geometry, and indeed, this style of pure mathematics, goes back to the earliest records of the subject as we would nowadays recognise it. The style emphasises the deduction of geometrical theorems from very simple axioms governing entities such as points and lines which otherwise have little to define them. Its canonical reference is unquestionably Euclid's *Elements* [38], possibly the most influential mathematical text ever written [6], and still a model for how pure mathematics is organised today, proceeding from axioms and definitions to great hierarchies of theorems.

Synthetic geometry is contrasted with *analytic geometry* or coordinate geometry. In analytic geometry, we solve geometric problems by translating them into systems of algebraic equations and then solving for unknowns. The use of algebra can be highly effective, but it is not always clear how to interpret the proof steps geometrically.

A synthetic proof, on the other hand, proceeds by introducing geometric entities which can be visualised as a diagram and reasoned about directly using simple principles, meaning that proof steps have a pleasing geometric interpretation. As we present our own proofs, we urge the reader to follow along with the help of pencil and straightedge. The subset of synthetic geometry we consider here means that no compasses are necessary!

1.1 The *Grundlagen der Geometrie*

The axioms which will form the basis for our geometry are taken directly from David Hilbert's *Grundlagen der Geometrie*. The text was chosen because in both modern mathematics and in the formal verification community, it has an impressive reputation. By the middle of the 20th century, it had been hailed as the most influential book on geometry in a hundred years [4], and by 1971 it had ten published German editions, the last of these being translated as the *Foundations of Geometry* (second edition) [42]. We follow the presentation in the English translation to the letter.

The text can be seen as the spiritual successor to the axiomatics of Euclid's own *Elements*. Euclid's text remains remarkable in what it accomplishes by reducing a wealth of geometric results to a handful of simple axioms and casting basic number theory in geometric terms, but Hilbert massively improves on the rigour.

Hilbert does away with Euclid's confusing list of pseudo-definitions, in which we are told, for example, that a point is "that which has no part". Instead, he lets the axioms exhaustively define everything we can know of points, thereby inviting us to leave our intuitions and presumptions at the door. In a famous remark, Hilbert went so far as to demand that all references to points, lines and planes in his text should be replacable with "mug", "table" and "chair" without affecting the logic of the arguments [103], thereby enforcing a principle from Pasch that all deductions must proceed without reference to the intuitive meaning of the terms involved [53]. Another way to put this is to say that Hilbert presents his axioms and their consequences without interpretation. If he is successful, all consequences should follow by the *form* of the axioms and not by their content, something we can test by seeing whether the axioms and proofs can be unambigously translated into formal logic.

1.2 Ordered Geometry

Hilbert has five groups of axioms to describe Euclidean geometry, but in this thesis, we are interested in a much more general *ordered geometry* [79]. The scope is defined by the first two of the five axiom groups, providing a more restrictive setting for doing proofs, where we lack a metric to talk about the distances between points, and we lack notions of angle or stipulations about parallel lines with which to discuss direction. We are without a sense of scale or orientation, but as Hilbert shows, we can still make

useful definitions.

And as we shall show, we can still demonstrate important results. The main one and, we daresay, the *fundamental result* of ordered geometry, is the Polygonal Jordan Curve Theorem. This theorem requires that any polygon divides the plane into exactly two connected regions. It appears as THEOREM 9 in the 10th edition of the *Grundlagen*, and all the previous theorems can be seen as setting down the preliminaries required to prove it. It will be the focal point of this thesis.

While the Polygonal Jordan Curve Theorem is relatively easy to prove when we have the full resources of topology and Euclidean geometry to hand, in the very general setting of ordered geometry, the proof is quite involved. In fact, it is reasonably certain that its first published proof by Veblen is invalid, and we shall argue our case for this in Chapter 8. Our proof, on the other hand, is on *far* firmer footing, for a major contribution of this thesis is its formal verification.

1.3 Verification

A formal verification consists in translating theorems and proofs to formal logic and then showing that all deductions are valid according to blind symbolic inference rules. The rules are so simple and few that it is easy to guarantee their validity, and thus we can guarantee the validity of any argument expressed in those rules by mechanically checking each step.

Partial verifications of Hilbert's axiomatics have been investigated by Dehlinger et al [21] and Meikle and Fleuriot [66]. What is particularly enticing about the work of Meikle and Fleuriot is their suggestion that there are logical gaps and unstated assumptions in some of Hilbert's prose proofs. We will explain those gaps in Chapters 3 and 5, where we shall try our best to justify them and vindicate Hilbert.

That said, we do not start from the assumption that Hilbert was infallible. Experience tells us that gaps are left open and logical errors easily made when doing synthetic geometry. The axioms place a severe handicap on the mathematician, preventing us initially from using geometric constructions and making observations that are so elementary that it is tempting to assume them implicitly and erroneously. With purely ordered geometry, we have an even more stringent handicap, and so we must be even more careful when trying to prove results. A diagram used to explore a proof can easily mislead by implying constraints that are not formally demonstrable, and so great care

must be taken to ensure proofs are valid. We might never be fully confident without a formal verification.

Now the formalisation, if not the verification, of the axiomatics and elementary consequences of Hilbert's *Grundlagen der Geometrie* was anticipated almost immediately. In his review of the text, Veblen [99] cites Peano, who had already translated Pasch's axiomatisation of projective geometry into a symbolic form. Peano's notation survives to this day and his ideas would inspire Russell [83] to produce the first major formal verification of elementary mathematics in *Principia Mathematica* [104].

But as Russell found out, verification can be very labour intensive, and when Poincaré saw the lengths Russell had to go just to verify that 1 is a number, he saw only "shackles" [40], and went so far as to call Peano's aims of verification "puerile" [39]. The criticisms were thankfully short-sighted. Verification is forcing itself on a reluctant world now that proofs have become so long and convoluted that they cannot always be verified by individual human readers [18], while the shackling pedantry required of Russell and Whitehead's research programme can be greatly alleviated with the help of machines.

1.3.1 Computer Assistance

Surprisingly, even the machine-assisted mechanical verification of Hilbert's text had been anticipated in reviews. Both Veblen and Poincaré mention mechanical logic machines that had been developed in the 19th century [3] with which it was hoped proofs could be automatically generated. It was early days, and they both overestimated the power of such early machines — one was limited to syllogisms — but by the mid-1950s, Herbert Simon had a logic machine which could automatically prove all the theorems in Russell's *Principia*, thereby paving the way for computer assisted verification which could relieve the poor human of the Herculean task of manually deriving the theorems.

The success of Simon's logic machine had Russell reflect on his manual verification as "wasted" effort [1], but humans were not to be made redundant. Instead, computer assistance empowers them to tackle more complex theories, such as Hilbert's.

A decade after Simon's logic machine came DeBruijn's AUTOMATH project and the first computer *assistant* for formally verifying some real mathematics. It was successfully used to verify Landau's classic text on real analysis [58, 98], and since then, com-

puter assisted verification has had some astounding successes. Take the Four Colour Theorem. This is a century old outstanding problem. Its first 1976 proof was assisted by inscrutable algorithms and was rightly viewed with suspicion, but the whole theorem has now been meticulously verified by Gonthier in an extremely robust verifier [25]. According to Hales, the verification makes the theorem one of the most well-established results in all of mathematics [30], and Gonthier went on to lead the project verifying the Feit-Thompson Theorem [26], a milestone in a potential verification of the classification of all finite simple groups. Finally, the verification of the outstanding four century old Kepler Conjecture has recently been accomplished [29].

1.3.2 Readable Verifications

A modern verification consists of code needed to drive and guide a computerised proof assistant, which can mechanically check the validity of inferences used in a symbolic proof. We want this code to be a possible substitute for synthetic prose proofs, and retain the same visual appeal. We want readers to be able to follow the steps of the verification, perhaps drawing diagrams, and see results emerge that match our geometric intuition.

We have therefore adopted the *declarative style* of verification which aims to be close to the "mathematical vernacular" [19], and which respects the logical progressions typical of synthetic geometry. Just as in synthetic geometry, we reason to our theorems by introducing geometric entities, obtaining configurations of points, lines and planes. We then use our axioms to derive interesting properties of the configurations.

This proves challenging, because of an observation made in the AUTOMATH project which has largely stood up: there is a wide gulf between mathematical proofs as they appear in the literature and verification code. The inferential leaps that a human mathematician makes when writing a prose proof are multiplied to many formal steps in the verification, and the term "DeBruijn factor" was coined for the multiplier.

The blow up can be seen in Meikle and Fleuriot's work [66], and our own earlier work [86], where many verification steps are needed for a single prose step, a fact which often obscures the intuition behind the proof. We did not find this acceptable. We were not prepared to sacrifice intuition on the altar of verification.

1.4 Contributions and Organisation

In the next chapter, we give an overview of the computerised proof assistant and the logic upon which we implemented all the ideas for this thesis. In Chapter 3, we present our formalisation of Hilbert's axioms, and try to explain why the axioms make verifications so much more long-winded than their prose counterparts. We also state and verify a theorem (Proposition 3.1) that Hilbert may have neglected and which, to our knowledge, has not been previously verified from these axioms.

In Chapter 4, we present new algorithms based on streams of proof trees, which define a general-purpose algebra for partitioning and searching domains of interest. We show how to tailor this algebra to theorem-proving, and then apply it specifically to Hilbert's axioms. Then, in Chapter 5, we show how the automation provided by our search algebra can greatly reduce the amount of code needed to verify theorems. In fact, we show that our verifications become almost structurally identical to Hilbert's prose, with each verification step formalising an inference in the prose. These theorems have been verified elsewhere, but we provide the first verifications which match the natural language proofs so closely.

In Chapters 6 and 7, we verify Hilbert's theorems leadings up to his statement of the Polygonal Jordan Curve Theorem. In the first of these chapters, we show how to carve out the natural numbers geometrically without needing the axiom of infinity, and we then show how our verification of Hilbert's theorems allows us to reduce problems of ordering on the line to inequalities of natural numbers.

Our remaining chapters cover the verification of the Polygonal Jordan Curve Theorem from the very weak axioms of ordered geometry. In Chapter 8, we give a mostly informal discussion of the theorem, before discussing problems with Veblen's first proof. In Chapter 9, we present our formalisation of the theorem, leaving the details of the verification to Chapters 10 and 11.

The proof and verification are both original, and because of the automation we provide in Chapters 4 and 6 and based on the evidence from Chapter 5, we hope that much of our verification code is structurally similar to what we would expect of a fully elaborated prose proof.

There is plenty of commentary about Hilbert's axiomatics throughout this thesis. This is necessary, because formalising and verifying proofs forces us to make decisions with a pedantry that even mathematicians might consider debilitating. The process of

teasing out logical distinctions and subtle paths of inference, and tracing dependencies between results, puts us in a strong position from which to commentate. So we include many observations about Hilbert's approach in stating theorems and proofs, some neutral, some critical, and we make these remarks with great precision and confidence: the application of formal verification and theorem proving in studying the logic of practised mathematics can be likened to the use of the microscope in studying biology.

Chapter 2

Background

In this chapter, we describe our chosen logic and our chosen verification tool, HOL Light [35]. This chapter is provided for readers who are familiar with quantifier logic but not theorem provers based on simple type theory. It contains very little in the way of new material, and so to assist readers wanting to skip topics familiar or otherwise irrelevant to them, we describe each section.

In §2.1, we give a brief overview of simple type theory. In §2.1.1, we give its formal definition.

In §2.2, we review the LCF approach to implementing interactive proof assistants, and our chosen proof assistant HOL Light. In §2.3, we describe the classical axioms which supplement the basic object logic of HOL Light. In §2.4, we describe some of the tools that are built on top of the basic proof assistant.

In §2.5, we introduce the idea of a declarative verification, and in §2.5.1, we discuss the Mizar Light language that we have used. In §2.5.2, we present previously unpublished material on some useful extensions and modifications to the Mizar Light language.

We direct the reader's attention to §2.6, in which we explain a few conventions on how we will use the words *theorem*, *formal theory*, *verification* and *formalisation* throughout the rest of the thesis.

2.1 Object Logic

The logic of our proof assistant is Church's Simple Theory of Types [12] or the simply-typed lambda calculus. This calculus is typically associated with the foundations of programming languages, but it was originally conceived as a foundation for logic [11],

one which avoided the paradoxes of the naïve set theories in a way that was less "artificial" than both ZF set theory and Russell's Ramified Type Theory [82].

The original lambda calculus turns out to be inconsistent (all terms are provably equal [15]), and ironically, Church fixed the problem in essentially the same way as Russell, though by the time he did so, Russell's ramified types had been simplified; hence, the *simple theory of types*.

There might be some concern that the use of lambda calculus is unfaithful to a mathematics which is supposedly based on first-order set theory, but practised verification usually favours typed higher-order logics. Russell and Whitehead's celebrated computer unassisted verification [104] was in a typed higher-order logic. The first non-trivial computer assisted verifications used DeBruijn's AUTOMATH [20] and a powerful extension of the simply typed lambda calculus. And according to Wiedijk's survey [106], eleven of the world's seventeen proof assistants were based on a higher-order logic as of 2006. Even those based nominally on untyped set theory such as Mizar [19] can be viewed as typed [107].

The definitions to follow include a standard simple extension to Church's original theory. In Church's formalism, there are theorems and proofs which hold for any substitution of their types, and which were stated as schemas. Rather than use schemas, we make substitutable types part of the syntax, by introducing *type variables*. Theorems (and terms generally) which feature these type variables are called *polymorphic* and replace the theorem schemas of the original calculus. This makes computerised provers based on simple type theory more efficient since polymorphic theorems have only one proof to verify while a schematic theorem must be verified for each type we want to apply it to. The expressive power of the logic is preserved since types can only be instantiated and never generalised.

2.1.1 Definitions

We now give a formal definition of the object logic.

2.1.1.1 Syntax

First of all, we introduce *types*, which can be understood as collections of values and which can be defined from an alphabet of type constants and an alphabet of type variables, such that

1. every type constant and every type variable is a type;
2. for types τ_1 and τ_2, we have that $\tau_1 \to \tau_2$ is a (function) type. Arrow composition is right-associative, so that $\tau_1 \to \tau_2 \to \tau_3$ parses as $\tau_1 \to (\tau_2 \to \tau_3)$.

Next, we have *terms* based on an alphabet of term constants and term variables. There is a typing relation $(:)$ associating terms and types. If we say that terms denote values, then we can say that the typing relation associates a term with the type of its value. The rules are:

1. every term constant and every term variable is a term;
2. for terms $f : \tau_1 \to \tau_2$ and $x : \tau_1$, we have that $f\ x$ is a term such that $f\ x : \tau_2$. These *combinations* are left-associative: $f\ g\ h = (f\ g)\ h$;
3. given a term variable $x : \tau_1$ and a term $y : \tau_2$, there is a term $\lambda x.\ y : \tau_1 \to \tau_2$. These *lambda abstractions* consume everything to the right, so $\lambda x.\ f\ x\ y$ parses as $\lambda x.\ ((f\ x)\ y)$.

The idea is that the type $\tau_1 \to \tau_2$ is inhabited by functions with domain τ_1 and codomain τ_2. A lambda abstraction in this type is then a function literal, and the notation $\lambda x.\ f\ x$ can be understood as the familiar mathematical notation $x \mapsto f\ x$.

All functions in simple type theory have arity 1. If we want an n-ary function with $n > 1$, we typically map a single argument to another function which expects the remaining $n-1$ arguments. So a two argument function from τ_1 and τ_2 to τ_3 is given type $\tau_1 \to \tau_2 \to \tau_3$. This is known as *currying*, and its pervasiveness explains why function application is left-associative in simple type theory. We want to write a two-argument function application as $f\ x\ y$, and not the more cumbersome $(f\ x)\ y$.

The syntax unifies several ideas from first-order logic. Rather than having a separate syntax for formulas and terms, we just declare that formulas *are* terms but in a designated type `bool` of truth values. We can then take predicates, sets and relations to be higher-order functions with codomain `bool`. Thus, a predicate is now just a function of type $\tau \to$ `bool` and a binary relation $\tau_1 \times \tau_2$ is a function of type $\tau_1 \to \tau_2 \to$ `bool`. Sets are represented by predicates (their characteristic functions), and by using functions from predicates to predicates, we can recover simple set-theoretic operations such as union and intersection.

Types restrict the set of terms that could otherwise be generated by combination and lambda abstraction, and prevent the paradox which motivated Russell and Whitehead's

logic. That paradox asks us to consider the set of all sets which do not contain themselves. When we understand a set X as some predicate $X : U \to \texttt{bool}$, and the assertion $X \notin X$ as the negation $\neg(X\ X)$, we would need to formalise Russell's paradoxical set as $\lambda X.\ \neg(X\ X)$. But this is impossible, since the X cannot be given a type consistent with rules 2 and 3. Russell's paradox is thus averted.

2.1.1.2 Calculus

The calculus for simple type theory used in HOL Light is based on the primitive type of truth values `bool` and the primitive (polymorphic) relation $(=) : \tau \to \tau \to \texttt{bool}$. With these, we can introduce the notion of *judgements* or *sequents*, which have the form

$$\{A_1 : \texttt{bool}, A_2 : \texttt{bool}, \ldots, A_n : \texttt{bool}\} \vdash C : \texttt{bool}.$$

We are to understand these sequents as saying that C is judged true in the context of assumptions $\{A_1, A_2, \ldots, A_n\}$. The sequents are linked by inference rules, which tell us how new sequents arise from existing sequents. In other words, we prove sequents, rather than propositions.

The rules of the calculus as used in HOL Light (Figure 2.1) are few, and so far as we know, admit only one redundancy: the transitivity inference rule is provided as an optimisation. Each rule is expressed as a horizontal line, with sequents below the line being derivable from sequents above. We omit all types, since these can always be inferred from the terms. In fact, proof assistants based on this logic, including HOL Light, will automatically infer the most general type of all terms [16].

There are two more rules to handle instantiations of free term variables and type variables. We have a rule to substitute terms $t_1, \ldots, t_n$ for term variables $x_1, \ldots, x_n$ (avoiding variable capture). We have another rule to substitute types $\tau_1, \ldots, \tau_n$ for type variables $\alpha_1, \ldots, \alpha_n$.

$$\frac{\Gamma[x_1, \ldots, x_n] \vdash p[x_1, \ldots, x_n]}{\Gamma[t_1, \ldots, t_n] \vdash p[t_1, \ldots, t_n]}\ \texttt{INST} \qquad \frac{\Gamma[\alpha_1, \ldots, \alpha_n] \vdash p[\alpha_1, \ldots, \alpha_n]}{\Gamma[\tau_1, \ldots, \tau_n] \vdash p[\tau_1, \ldots, \tau_n]}\ \texttt{INST\_TYPE}.$$

We omit the formal definitions of substitution here. We just mention that λ *binds* free variables just like a quantifier does.

$$\frac{}{\vdash x = x}\ \texttt{REFL}$$

$$\frac{\Gamma \vdash s = t \quad \Delta \vdash t = u}{\Gamma \cup \Delta \vdash s = u}\ \texttt{TRANS}$$

$$\frac{\Gamma \vdash x = y \quad \Delta \vdash f = g}{\Gamma \cup \Delta \vdash f\ x = f\ y}\ \texttt{MK\_COMB}$$

$$\frac{\Gamma \vdash s = t}{\Gamma \vdash (\lambda x.\ s) = \lambda x.\ t}\ \texttt{ABS}, \quad \text{for } x \text{ not free in } \Gamma$$

$$\frac{}{\vdash (\lambda x.\ t)\ x = t}\ \texttt{BETA}$$

$$\frac{}{\{A\} \vdash A}\ \texttt{ASSUME}$$

$$\frac{\Gamma \vdash P = Q \quad \Delta \vdash P}{\Gamma \cup \Delta \vdash Q}\ \texttt{EQ\_MP}$$

$$\frac{\Gamma \vdash P \quad \Delta \vdash Q}{(\Gamma - \{Q\}) \cup (\Delta - \{P\}) \vdash P = Q}\ \texttt{DEDUCT\_ANTISYM\_RULE}$$

Figure 2.1: Inference rules

2.1.2 Higher-order Logic

With only equality and lambda abstraction, it might be surprising that the calculus embeds a full (intuitionistic) higher-order logic (hereafter *HOL*). The definitions of all connectives and quantifiers are given in Table 2.1, ordered by precedence rather than definitional dependency.

The logic is higher-order since the variable x in both quantifiers is polymorphic. Thus, we can quantify over any type, including predicates and functions. We can also quantify over truth values, a fact which is exploited in the definitions of disjunction and the existential quantifier.

All encodings are intuitionistic. In particular, notice that $P \Longrightarrow Q$ is not defined as $\neg P \vee Q$, and disjunction and existential quantification are not defined via the DeMorgan correspondences, as they might be in a classical logic.

(p,q)	$\lambda f.\ f\ p\ q$	Pairing
$fst\ P$	$P\ (\lambda x.\ \lambda y.\ x)$	First projection
$snd\ P$	$P\ (\lambda x.\ \lambda y.\ y)$	Second projection
$\top$	$(\lambda x.\ x) = \lambda x.\ x$	Truth
$\bot$	$\forall P.\ P$	Absurdity
$\neg P$	$P \iff \bot$	Negation
$P \wedge Q$	$(P,Q) = (\top,\top)$	Conjunction
$P \vee Q$	$\forall R.\ (P \implies R) \implies (Q \implies R) \implies R$	Disjunction
$P \implies Q$	$P \iff P \wedge Q$	Implication
$P \iff Q$	$P = Q$	Equivalence
$\forall x.\ P\ x$	$(\lambda x.\ P\ x) = \lambda x.\ \top$	Universal
$\exists x.\ P\ x$	$\forall R.\ (\forall x.\ P\ x \implies R) \implies R$	Existential
`let` $x = t$ `in` y	$(\lambda x.\ y)\ t$	Local variable definition

Table 2.1: HOL basics

2.2 Proof Assistant

HOL Light is implemented in the Ocaml programming language. The roots of both go back to the original statically typed functional language ML [71], and the LCF tradition [69].

2.2.1 Edinburgh LCF

If natural languages serve as the metalanguages for defining object logics for human proof checkers, then ML serves as the metalanguage for defining the object logics of *computerised* proof checkers. Accordingly, when we define the syntax of an object logic, ML understands "inductive data type" where we understand "inductive definition", and ML understands "function from sequents to sequent" where we understand "inference rule".

The use of a programming language can clarify a lot. Consider the definition of the `ABS` inference rule in Figure 2.1, where we say "for x not free in Γ", a qualification which is absent from the definition of say, `BETA`. The ML code makes the difference here very clear: the function implementing the inference rule `ABS` takes a variable x as

an additional argument, while that for `BETA` does not.

Sequents and inference rules in HOL Light, as in all LCF systems, are implemented in ML as a *trusted kernel*. To gain the trust, a user can carefully peruse the kernel's modest 672 lines of ML, especially the 160 or so lines of code needed to deal with free variable substitutions, instantiations, and testing whether two terms are equivalent up to a consistent renaming of bound variables (formally, testing for "α-equivalence").

This last bit of the code is tricky to reason about, just as it is tricky to define and reason about free-substitutability in basic proof theory. In fact, two bugs were identified and fixed in this part of the HOL Light code circa 2003 [9]. No other bugs have been found since, and in 2006, Harrison presented a formal verification that this part of the kernel (in fact, the whole kernel and axioms minus only the definitional facilities described below) is sound and consistent *from within HOL Light itself*![1][36] The possibility of any further bugs is now vanishingly small, and so we can put extremely high confidence in our verified proofs.

User code, again written in ML, calls the inference rules of the kernel in order to construct sequents and thereby mechanically verify theorems. All the mechanical verifications mentioned in this thesis consist of such code. However, it is rare that we use the kernel directly. Having a full programming language available to user code means that LCF systems typically offer powerful derived rules, embedded proof languages, and fully automated tools to assist in constructing sequents. HOL Light is no exception, and we discuss some of its offerings in §2.4, and in Chapter 4, we describe our own forward search tool.

We are not expected to exploit the implementation of sequents and inference rules, and indeed *cannot*. Unlike the term data-type (`term` in HOL Light), the sequent data type (`thm`) is made *abstract* to user code. ML's strong type system guarantees that user code cannot break this abstraction. Any sequent constructed in user code is a sequent that was ultimately constructed legitimately via the kernel's inference rules.

2.2.2 Additional Functionality

The HOL Light kernel extends the simple theory of types by allowing us to add new axioms (see §2.3) and add new basic term and type definitions. *Basic definitions* cannot contain free variables on their right hand sides, nor can they be recursive. Thus, they

[1] To be consistent with Tarski's Theorem, the soundness proof requires an additional axiom that there exists a "large" cardinal (large from the perspective of HOL but not from ZF set theory)

always yield a conservative extension of the logic: we could potentially substitute the right hand sides of any definitions through all terms without affecting the validity of the derivations.

HOL Light also extends the simple theory of types by allowing one to define a new type of values, the *abstract* type, in bijection with a non-empty subset of an existing type, the *representation* type. This feature is particularly useful when forcing derived definitions to be used in a way which respects an abstraction.

For instance, in Chapter 7, we define both *rays* and *half-planes* as abstract types in bijection with representative point-sets satisfying appropriate constraints. For rays, we define an *endpoint* in terms of the representative point-sets, and can thus use expressions such as "endpoint of a ray". However, since the types are abstract, the expression "endpoint of a half-plane" is not well-typed, even though both rays and half-planes have the same representation. The type `ray` enforces the intended abstraction. As an added benefit, because HOL Light types can always be mechanically reconstructed from expressions, abstract types can eliminate the need for some side-conditions by pushing the constraints on their representatives into an automatically inferred type.

The combination of type definitions and polymorphism leads to another desirable but simple extension. Types can be defined from polymorphic definitions, such as lists which are polymorphic in their element type. In this case, the type variables in the definition become the arguments of a *type constructor*. The extension replaces the two rules of the type syntax with the single rule

> for every natural number n (the *arity*), type constructor T and types $\tau_1, \tau_2, \ldots \tau_n$, we have that $T\ \tau_1\ \tau_2\ \ldots\ \tau_n$ is a type.

Type constants are now just type constructors with arity 0, function types are formed from a type constructor $(\rightarrow)$ of arity 2, and the type of lists is a type constructor `List` with arity 1. For readability, we write $(\rightarrow)\ \tau_1\ \tau_2$ as $\tau_1 \rightarrow \tau_2$ and `List` τ as $[\tau]$.

2.3 Classical Logic

HOL Light allows any formula A to be permanently asserted as an axiom, giving the sequent $\vdash A$. We shall use this facility in the next chapter to define the axiomatics of Hilbert's geometry. The facility is also used in the standard distribution of the system to make HOL a classical logic.

A standard classical axiom added to HOL Light is the axiom of extensionality, or the η-reduction axiom.

$$\vdash \forall f.\ (\lambda x.\ f\ x) = f.$$

With it, one can show that $f = g$ precisely when $f\ x = g\ x$ for arbitrary x. Thus, functions are equal when they agree on their outputs, and sets are equal precisely when they have the same members.

The propositional fragment of HOL becomes classical with the introduction of the term ε, whose sole axiom is:[2]

$$\vdash \forall P.\ \forall x.\ P\ x \implies P\ (\varepsilon x.\ P\ x).$$

We are to understand $\varepsilon x.\ P\ x$ as *the arbitrarily chosen x satisfying P*. This is the full axiom of choice, used frequently as a definitional tool. It should be contrasted with the weaker ι binder, which yields expressions $\iota x.\ P\ x$ to be read as *the uniquely specified x satisfying P*. The distinction is discussed further in §6.5.

The axiom for ε clarifies how simple type theory handles undefined or non-referring terms. Consider what happens when P is unsatisfiable. In this case, $\varepsilon x.\ P\ x$ is still, in a sense, well-defined. From the perspective of the logic's model theory, all terms, even peculiar ones such as $\varepsilon x.\ P\ x$ where P is unsatisfiable, must refer. But from a proof-theoretic point of view, since the condition on the axiom can never be discharged for this particular P, nothing interesting can ever be shown true of $\varepsilon x.\ P\ x$. The only sequents we can derive of it are logical truths, and if we take Wittgenstein at his word in the *Tractacus* [109], we might say that when all we can derive are logical truths, we have nothing. It is in this sense that we can formalise the notion of undefined terms and the undefined values of partial functions in HOL. Each one is just $\varepsilon x.\ \bot$.

The axiom of choice is sufficient to derive the law of excluded-middle, by exploiting the fact that ε terms with equivalent bodies are equal. In particular, we consider $u = \varepsilon x.\ x \vee P$ and $v = \varepsilon x.\ \neg x \vee P$. Since we have $\top \vee P$ and $\neg\bot \vee P$, we obtain by the axiom of choice $(u \vee P) \wedge (\neg v \vee P)$ which is equivalent to

$$(u \wedge \neg v) \vee P. \tag{2.1}$$

Now on the hypothesis of P, the terms u and v simplify to $(\varepsilon x.\ x \vee \top) = \varepsilon x.\ \top$ and $(\varepsilon x.\ \neg x \vee \top) = \varepsilon x.\ \top$ respectively. Thus, they become equal. We then have

$$P \implies u = v \implies \neg(u \wedge \neg v),$$

[2]Here, ε acts as a binder, like λ, $\forall$ and $\exists$, but this is syntax sugar. In reality, ε is a term of type $(\tau \to \texttt{bool}) \to \tau$.

or alternatively, $u \wedge \neg v \implies \neg P$. We then conclude from (2.1) that $\neg P \vee P$.[3]

We end this subsection by introducing a final piece of core syntax. The axiom of choice allows us to encode a conditional operator as follows:

$$\texttt{if}\ b\ \texttt{then}\ x\ \texttt{else}\ y \equiv \varepsilon z.\ (b \implies z = x) \wedge (\neg b \implies z = y).$$

2.3.1 Axiom of Infinity

The final axiom in HOL Light is the axiom of infinity. We will discuss this axiom in some detail in Chapter 6. For now, we will only say in advance that it is provably redundant when we have Hilbert's first two groups of axioms. We have therefore deleted it, an act we are tempted to believe would appease a later sceptical Hilbert, who would come to reject infinite sets as a foundational element of mathematics [43]: "[the infinite] neither exists in nature nor provides a legitimate basis for rational thought."

2.4 Verification Tools

Outside of the relatively tiny HOL Light kernel is a huge collection of verification tools written in user code. In many cases, we do not care how these tools work, or whether they even contain bugs, since bugs cannot penetrate the boundary of the kernel. We only care that the tools generate the sequent we want.

2.4.1 Tactics

HOL Light implements the LCF tactic system [70], in which we understand the process of constructing a sequent as the solving of a goal by breaking it into subgoals. We will need to briefly discuss the implementation of tactics for §2.5, since we make use of some of the lower level details.

As far as tactics go, a *goal* is a pair consisting of a goal formula together with *hypotheses*. The purpose of a tactic is to input a goal and produce zero or more subgoals. These are then collected onto a goal stack, which acts like an agenda of problems that the user must solve. The user's aim is to apply tactics until the agenda is empty. A sequent is then automatically proven by following the trace of the tactic steps back to the goal.

[3] The propositional inferences here are all intuitionistically valid.

2.4.2 Fully Automated Procedures

Some tactics in HOL Light are decision procedures or otherwise fully automated verification tools. Generally, these are used to solve a goal outright; that is, without generating any new subgoals. Among HOL Light's decision procedures are those for checking tautologies, for deciding problems in linear arithmetic, and for computing ideal elements via Gröbner bases [8].

More recently, HOL Light has been able to make use of external theorem proving programs. Eekelen et al [57] have implemented HOL Light code to take proof certificates output by automated theorem provers for universally quantified propositional logic. These certificates are then interpreted in ML and translated into the appropriate HOL Light inference rules needed to produce corresponding HOL Light sequents. Similar progress has been made in HOL4 [56] and Isabelle [68].

2.5 Declarative Proof

Proof assistants input verifications in languages of broadly two forms: declarative and procedural. In a procedural language, the user employs tactics to compose automated tools in order to produce the desired sequents. Different tactic languages have their own styles and idioms, but they usually support both forward reasoning from the premises of an argument to its conclusion, and backward reasoning, breaking down the goal conclusion into simpler subgoals. Always the focus is on procedural transformations rather than logical formulas, which are sometimes entirely absent from the procedural proof texts.

Declarative proof languages on the other hand, inspired by *Mizar* [19], attempt to imitate the style of ordinary mathematics. The verifications show the flow of argument as *steps* which introduce intermediate formulas, with branches at subproofs and case-splits. The proofs begin by stating assumptions, with steps always being justified either from known lemmas or from prior steps. The focus is on *what* the logical relations between formulas are, rather than *how* the internal state (the goal stack in the case of tactics) is transformed to represent such relations. This style of proof assistant relies heavily on automation, guided by how the user breaks the proof down into intermediate formulas.

The declarative style was a natural choice for verifying synthetic geometry. As we

explained in Chapter 1, we want our verifications to follow the structure of synthetic prose arguments, building diagrams and reasoning about their properties.

2.5.1 Mizar Light

Mizar Light, developed by Wiedijk [105], is a declarative style language embedded in HOL Light and inspired by the primitives of the declarative proof assistant, Mizar [19]. We give an overview of its primitives in Figure 2.2. With the exception of `using`, we have emphasised a declarative semantics: rather than describing *how* each primitive affects the state of the prover, we describe *what* each primitive asserts at a given point in a script.

As noticed by Harrison [33], the operational semantics of these primitives can be given in terms of tactics and simplified goals. The tree of goal stacks becomes a tree of subproofs and case-splits. The hypotheses of each goal become the intermediate lemmas. Each step becomes a tactic which drives the verification forward. Here, for instance, is a typical step one might read in one of our Mizar Light verifications:

`consider` *P* `such that` ¬`on_line` P a `by` (I, 2),(I, 3.1).

This `consider` step translates to a tactic which introduces a subgoal with formula ∃*P*. ¬`on_line` *P a*. The goal is solved outright using the step's *justification*. By default, steps are justified by HOL Light's generic `MESON` tactic [65], but additional tactics can be composed with the `using` keyword. The justification tactic usually needs some help to solve its goals, and in declarative verification, we name justifying sequents using the keyword `by`. In this example, we have added some sequents (I, 2 and I, 3.1) which are passed directly to `MESON`.

All Mizar Light primitives are ordinary ML functions, and the Mizar Light language is really just a combinator language [94]. This has been useful to us, since it is almost trivial to modify and extend Mizar Light by writing new combinators.

That combinators make it easy to extend a system is a feature we find particularly attractive. Tactics themselves are implemented as combinators, which makes it easy for users to write their own (we describe a few of ours in Chapters 4 and 6). Tactics also compose algebraically. There are tensoring operators, sums, identities and a zero. We have tried to keep to the spirit of algebraic combinator languages in our own automation, described in Chapter 4.

Primitive	Meaning
`theorem` *P*	Begins a proof of *P*.
proof *proof*	Asserts *proof* as a justification for the current step.
assume *P*	Asserts *P* as a justifiable assumption at this point.
so	Refers to the previous step as justifying the current step.
`have` *P*	Asserts *P* as derivable at this point.
thus *P*	Asserts *P* as derivable at which point the (sub)theorem is justified.
`hence` *P*	As `so thus` *P*.
take *var*	Identifies *var* as the witness for the (sub)theorem.
fix *vars*	Establishes *vars* as fixed but arbitrary variables.
consider *vars* `st` *P*	Introduces *vars* witnessing *P*.
from *steps*	Refers to proof steps *steps* as justifications for the current step.
by *thms*	Refers to previously established theorems *thms* as justifications for the current step.
using *tactics*	Augments the justification of this step with *tactics*.
per cases *cases*	Begins a case-split into *cases* with their proofs.
suppose *P*	A syntactic marker to identify an assumption *P* in a *case* analysis.
otherwise *proof*	Indicates that the (sub)theorem *thm* can be established by *proof*, which derives a contradiction from $\neg thm$.
`set` *bindings*	Introduces local variable bindings.
qed	Asserts that the (sub)theorem is justified at this point.

Figure 2.2: An overview of Mizar Light

2.5.2 Extending Mizar Light for Interactivity

Wiedijk's basic combinators are based on the original Mizar system, a batch prover, and in this spirit, Mizar Light verifications are written in their entirety and then evaluated in one. We found this undesirable, firstly, because the error reporting is not rich enough to show where errors occur in the case of a failed verification. Secondly, we chose to implement our automation (described in Chapter 4) so that it ran concurrently as we developed our verifications. Here, the tool works best when it can exploit our idle time when working *interactively* as opposed to *batch* mode.

The problem lies with case-splitting. Here are the original combinators at work in an extract of one of Wiedijk's example verifications (the details of which are not important):

```
...
have "∀p1. ∀p2. ∃l. p1 ON l ∧ p2 ON l" at 9
proof
 [fix ["p1:Point"; "p2:Point"];
  per cases
   [[suppose "p1 = p2";
     qed from [0] by [LEMMA1]];
    [suppose "¬(p1 = p2)";
     qed from [1]]]];
...
```

Above, we have a case-split on formulas `p1 = p2` and `¬(p1 = p2)`. The steps in each case are collected in lists, which makes for a neatly structured verification, where trees of subproofs and case-splits are reflected by ML data-structures. However, the steps of an interactive verification are supposed to be applied linearly, one-by-one, traversing an implicit tree. Here is what we prefer to write at the interpreter (> marks the ML prompt):

```
> have "∀p1. ∀p2. ∃l. p1 ON l ∧ p2 ON l" at 9
> proof
> fix ["p1:Point"; "p2:Point"]
> per cases
> suppose "p1 = p2"
> qed from [0] by [LEMMA1]
> suppose "¬(p1 = p2)"
> qed from [1]
```

2.5.2.1 Interactive Case-splits

Case-splits are ultimately justified by proving a disjunction of all considered cases. However, in Mizar-style verifications, as is common in ordinary mathematical proofs, the particular disjunction is never stated explicitly. Consider again the extract of Mizar Light code:

```
per cases
 [[suppose "p1 = p2";
   qed from [0] by [LEMMA1]];
  [suppose "¬(p1 = p2)";
   qed from [1]]]];
```

Here, there are two cases being considered: `p1 = p2` and `¬(p1 = p2)`. The disjunction which justifies them as exhaustive

```
p1 = p2 ∨ ¬(p1 = p2)
```

does not appear in the verification. Instead, it is assembled by the `per cases` step by gathering the `suppose` formulas, before the tactics for each case are ever applied. This is possible, because `per cases` takes the full list of cases, from which the disjunction can be assembled. But this strategy will not work if we are to linearise the subproofs and apply each step interactively, since the full disjunction will not be known until all cases are interactively solved.

Harrison's original Mizar mode for HOL Light had better support for interactive case-splitting [33]. The drawback appears to be that it relies on potentially invalid tactics, being tactics which succeed but from which a final verification cannot be recovered. Our implementation is different. We use two functions `case` and `end`. The `case` function is used to introduce a new case formula P. It then generates two subgoals, the first with P as hypothesis, and the second with $\neg P$ as hypothesis. The `case` step, therefore, has performed a case-split on $P \vee \neg P$. The user must first prove the goal on the hypothesis of P. Once the goal is solved, the one remaining goal will have $\neg P$ as its hypothesis.

The user now proceeds by introducing the next case, using the `case` function again with a new formula, say Q. Two subgoals are again generated, one with Q and the other with $\neg Q$ as hypothesis.

By the time the user has considered and proven all cases, the one remaining subgoal will have the negations of every considered case in its hypotheses. If the cases are

exhaustive, the negations will entail a contradiction. This is where the `end` step is used. It will automatically take all the negated cases, identifying them by a case-label `Case` in the goal stack, and use them as a justification for $\bot$.

Suppose, for example, that we have a sequent $\vdash P \vee Q \vee R$ assigned to ϕ and suppose that a goal with term G can be solved on each of the hypotheses P, Q and R using just the implicit automation built into Mizar Light. Then we can write the verification:

```
> theorem "G"
> case "P"
>  qed
> case "Q"
>  qed
> case "R"
> end by ϕ
```

The resulting tree of goal stacks is depicted in Figure 2.3. The final `end` step is justified since

$$P \vee Q \vee R, \neg P, \neg Q, \neg R \vdash \bot.$$

This approach only uses valid tactics. Whenever the case-splitting is not exhaustive, the `end` step will immediately fail.

Returning to the example verification, our functions `case` and `end` allow us to write

```
> lemma "∀p1. ∀p2. ∃l. p1 ON l ∧ p2 ON l" at 9
>  fix ["p1:Point"; "p2:Point"]
>  case "p1 = p2"
>    qed from [0] by [LEMMA_1]
>  case "¬(p1 = p2)"
>    qed from [1]
>  end
```

2.5.3 Concluding Remarks

Allowing interactive case-splitting worked well in practice. We would write our verifications interactively, and when completed, package them up as batch verifications. The translation between flattened verification and the normal batch `per cases` combinator was always straightforward.

Further modifications to Mizar Light are described in Chapter 4.

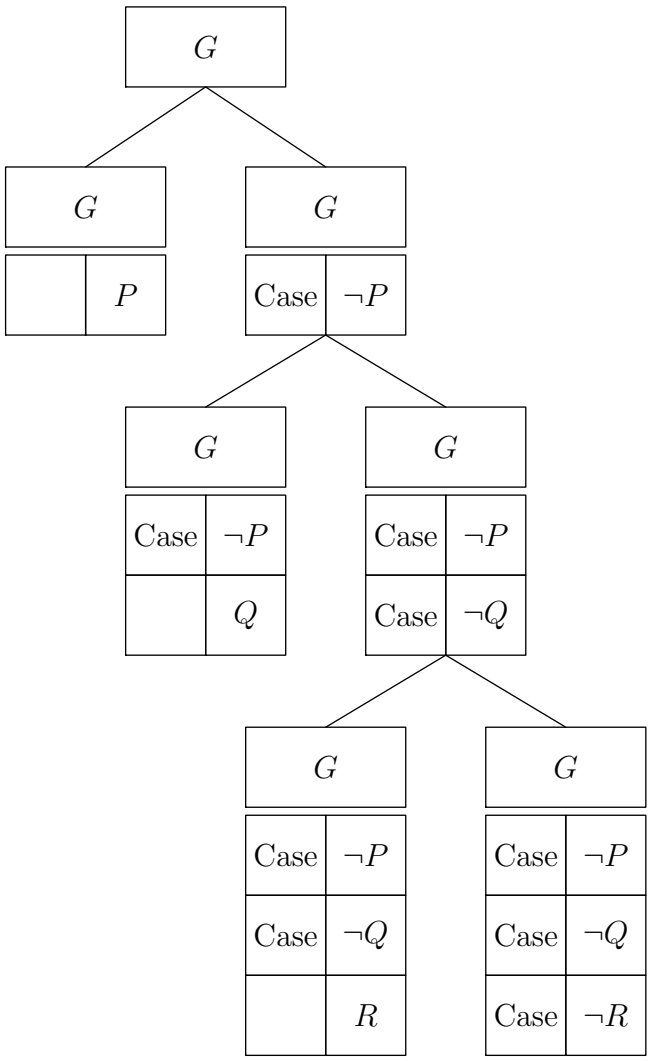


Figure 2.3: Case-splitting proof tree

2.6 Conventions

As with any thesis of this sort, there are certain ambiguities which arise from overloading and which ought to be clarified. We will adopt some terminology.

We use *formalise* to refer to the translation of natural language statements into the term language of HOL Light. We then reserve the words *theorem* and *lemma* for sequents $\vdash t$ we have produced in HOL Light where t formalises a natural language theorem. Thus, when we talk of theorems and lemmas from here on, we are referring to results that have been mechanically verified. If we want to talk more generally about sequents $\Gamma \vdash t$ which appear during proofs, we shall use the word *sequent* specifically. If we want to talk about non-formalised theorems, we shall use various other terms, and

when we want to refer to Hilbert's named results in particular, we shall use *THEOREM* in uppercase, as he does in the *Grundlagen*.

The word *verification* will be used for the ML used to construct a theorem. When we say that we *verify* a proposition, we mean that we have a verification of a theorem which formalises the proposition.

The term *formal definition* shall be used to refer to types and terms introduced using the definitional facilities of HOL Light. This will lead to new equational theorems between the left and right hand sides of the definition which we shall notate by $\vdash_{def} x = t$.

Finally, the term *(formal) theory* will be used to refer to a logically coherent body of formal definitions and theorems.

Throughout the thesis, we shall give sample verifications. These depict actual Ocaml code, but for readability, we elide various pieces of syntax such as brackets and semi-colons. We also introduce our own syntactic sugar, shown in the following table.

Notation	Translation
$\neg$	`~`
$\wedge$	`/\`
$\vee$	`\/`
$\Longrightarrow$	`==>`
$\Longleftrightarrow$	`<=>`
$\forall$	`!`
$\exists$	`?`
λ	`\`
$P \neq Q$	`~(P=Q)`
α, β, γ	`'a, 'b, 'c`
$x, y \in A, B$	`x IN A /\ y IN A /\ x IN B /\ y IN B`

The Mizar Light language uses expressions of the form `at` *m* to label an intermediate result with the number *m*. This number is then used by the combinator `from` to justify later steps by cross-referencing.

We have changed the implementation slightly so that the `at` combinator takes a list. Now an expression such as `at [`*m*`,` *n*`,` *p*`]` can be used to label the individual conjuncts (in this case, three of them) of an intermediate step, each of which can be cross-referenced separately. In the sample verifications in this thesis, we shall elide the combinator `at` and instead set *m*, *n* and *p* as right-justified labels in the verification.

Many of our ideas evolved during the verification development, and, as of writing, there are legacy naming conventions which need to be refactored and brought into line with the present thesis. For now, we can only claim that the verifications reproduced in the thesis are α-equivalent to the originals.

Chapter 3

Axiomatics

Hilbert has only the briefest introduction to the *Grundlagen der Geometrie*, before diving in with a declaration of his primitive notions and then laying out his five groups of axioms. In this chapter, we discuss the formalisation of the first two groups, a very weak axiomatic environment under which it is nevertheless possible to verify the Polygonal Jordan Curve Theorem. We also discuss the verification of a few of the elementary theorems, and how the first group of axioms in particular feature in the rest of our verifications.

3.1 Primitives

Hilbert opens his axiomatics by declaring three sets of primitive objects: a set of objects called *points*, a set of objects called *lines* and a set of objects called *planes*. These sets are *abstract*. All we can know about their inhabitants is what is specified by Hilbert's axioms.

That we call these abstract objects *points*, *lines* and *planes* can be thought of as mere documentation. It has no real significance to the formal theory, and this lack of significance is something that Hilbert and Pasch regarded as fundamental to rigour in geometry [103]. As Hilbert was known to remark, it would serve just as well to call the inhabitants of the three sets "mugs", "tables" and "chairs" [53].

This is the modern axiomatic method, and it is a noble sentiment if we hold rigour in such high esteem. But it is one thing to say that it is possible to run the substitution and quite another to carry it out. We will not be evaluating the matter, but we would conjecture that it would be very difficult for a human to follow the steps of Hilbert's

arguments if they were literally rendered in terms of mugs, tables and chairs.

This might explain the labour involved with verified mathematics. Our computers carry out Pasch's idea of stripping away all interpretation. They might as well be reading about mugs, tables and chairs. They see nothing but abstract symbols, and must validate the arguments without the help of intuition. If this is too much for a human, then it is quite something to expect of a machine.

Having introduced points, lines and planes, Hilbert goes on to declare that "[t]he points are also called the *elements of line geometry*; the points and the lines are called the *elements of plane geometry*; and the points, lines and planes are called the *elements of space geometry* or the *elements of space*." These comments do not appear to have any importance to our verification, and we are happy to ignore them, and all others like them, without worrying about jeopardising our aims of following Hilbert to the letter. Even when they take the form of definitions, we treat them as mere *documentation*, useful signposts for readers, but no more.

In general, we only introduce formal definitions when they identify abstractions we intend to use in verifications.

3.2 Group I

3.2.1 Incidence Relations

Following the abstract sets of points, lines and planes, Hilbert introduces a primitive relation *lie*, whose axioms are intended to characterise it as an incidence relation. With it, we can say that a point *lies* on a line, or that a point *lies* on a plane.

In a foreword to the text, Professor Goheen says that there must, in fact, be *two* relations. It is not clear to us why. Perhaps Goheen is taking his perspective from first-order logic. In this case, the sets would most naturally be represented by distinct and disjoint *sorts*, and thus, we would need two *lie* relations, one for each sort.

There is often an implicit assumption on sorts, namely that they are inhabited. This assumption is removed in *free-logics*, usually for philosophical concerns, and often at the expense of breaking standard inference rules (see Mendelson's classic text [67]).

Whether or not Hilbert was making the assumption that his sets were inhabited is unclear. Fortunately, we do not need to be too concerned, since the assumption is not needed. To formally settle this, we consider a formalisation that Goheen perhaps

$$\begin{aligned}
&\vdash \texttt{Group1}\ (\texttt{point} : \tau \to \texttt{bool}, \texttt{line} : \tau \to \texttt{bool}, \texttt{plane} : \tau \to \texttt{bool},\\
&\qquad \texttt{lie} : \tau \to \tau \to \texttt{bool})\\
&\iff (\forall A.\ \forall B.\ \texttt{point}\ A \wedge \texttt{point}\ B \wedge A \neq B \implies \exists a.\ \texttt{line}\ a \wedge \texttt{lie}\ A\ a \wedge \texttt{lie}\ B\ a)\\
&\quad \wedge (\exists A.\ \exists B.\ \exists C.\ \texttt{point}\ A \wedge \texttt{point}\ B \wedge \texttt{point}\ C\\
&\qquad \wedge \forall a.\ \texttt{line}\ a \implies \neg(\texttt{lie}\ A\ a \wedge \texttt{lie}\ B\ a \wedge \texttt{lie}\ C\ a))\\
&\quad \wedge (\forall A.\ \forall B.\ \forall C.\ \texttt{point}\ A \wedge \texttt{point}\ B \wedge \texttt{point}\ C\\
&\qquad \wedge (\forall a.\ \texttt{line}\ a \implies \neg(\texttt{lie}\ A\ a \wedge \texttt{lie}\ B\ a \wedge \texttt{lie}\ C\ a))\\
&\qquad \implies \exists \alpha.\ \texttt{plane}\ \alpha \wedge \texttt{lie}\ A\ \alpha \wedge \texttt{lie}\ B\ \alpha \wedge \texttt{lie}\ C\ \alpha)\\
&\quad (\exists A.\ \exists B.\ \exists C.\ \exists D.\ \texttt{point}\ A \wedge \texttt{point}\ B \wedge \texttt{point}\ C \wedge \texttt{point}\ D\\
&\qquad \wedge \forall \alpha.\ \texttt{plane}\ \alpha \implies \neg(\texttt{lie}\ A\ \alpha \wedge \texttt{lie}\ B\ \alpha \wedge \texttt{lie}\ C\ \alpha \wedge \texttt{lie}\ D\ \alpha)
\end{aligned}$$

$$\vdash \texttt{Group1}\ \texttt{point}\ \texttt{line}\ \texttt{plane}\ \texttt{lie} \implies (\exists A.\ \exists a.\ \exists \alpha.\ \texttt{point}\ A \wedge \texttt{line}\ a \wedge \texttt{plane}\ \alpha)$$

Figure 3.1: Points, lines and planes exist

neglected: we will represent each of Hilbert's three sets by a predicate, and consider relativising Hilbert's axioms to these predicates. This is, in effect, the embedding of a free-logic in classical logic. It has the additional benefit of allowing us to consider just one primitive incidence relation on one primitive sort.

Before we give our formalisation, we mention that we shall be adopting Hilbert's convention throughout this work, that points are denoted by uppercase Roman, A, B, C, P, Q, R, X, Y, Z, and so on. Lines are denoted by lowercase Roman a, b, c. And planes are denoted by Greek α, β, γ.

The formalisation of this single-sorted geometry is given in Figure 3.1. We have formalised four of Hilbert's incidence axioms (I, 1, I, 3.2, I, 4.1 and I, 8) as conditions on the predicate sets `point`, `line` and `plane` and the single relation `lie`. These predicates are polymorphic of a single type variable τ, which formalises the single sort for all our geometric entities. We have verified that any four objects satisfying these conditions are such that the three predicate sets are inhabited.

3.2.2 Axioms and Formalisation

With the verification showing that Hilbert's primitive sets must be inhabited, we are free to use what we regard as a more natural formalisation of Hilbert's axioms, the one provided independently by Meikle and Fleuriot's [66] and Dehlinger et al [21]. We declare three primitive types for points, lines and planes. The implicit constraint that these types are inhabited is permissible based on the verification of the previous subsection, and we can therefore take it as faithful to Hilbert's intended interpretation.

We then consider two incidence relations: one tells us whether points lie on a line and the other whether points lie on a plane. This gives a more readable formalisation than that of Figure 3.1, since we can drop the relativising predicates. It also improves type-safety: HOL Light can reject axioms which do not use the primitive relations in sensible ways, and it removes the possibility of nonsense expressions such as "a plane lies on a point."

We now give Hilbert's incidence axioms as they appear in the second edition of the *Foundations of Geometry* [42], translated from the tenth edition of the *Grundlagen der Geometrie*.

> I, 1 *For every two points A, B there exists a line a that contains each of the points A, B.*
>
> I, 2 *For every two points A, B there exits [sic] no more than one line that contains each of the points A, B.*
>
> I, 3 *There exist at least two points on a line. There exist at least three points that do not lie on a line.*
>
> I, 4 *For any three points A, B, C that do not lie on the same line there exits [sic] a plane* α *that contains each of the points A, B, C. For every plane there exists a point which it contains.*
>
> I, 5 *For any three points A, B, C that do not lie on one and the same line there exists no more than one plane that contains each of the three points A, B, C.*
>
> I, 6 *If two points A, B of a line a lie in a plane* α *then every point of a lies in the plane* α*.*
>
> I, 7 *If two planes* α*,* β *have a point A in common then they have at least one more point B in common.*
>
> I, 8 *There exist at least four points which do not lie in a plane.*
>
> (p. 4)

These axioms have undergone substantial revision since the first edition, being re-ordered, with some combined, some split and redundancies deleted. The fact that there

are redundancies in the first place goes to show that Hilbert's later claim that his axioms are independent was never *fully* investigated in the *Grundlagen der Geometrie*. In fact, only a few interdependencies were ever considered.

The axioms are formalised in Figure 3.2 and are asserted in HOL Light. We just give some supplementary discussion.

Hilbert's Axioms I, 3 and I, 4 each contain two distinct claims. We have not identified any interesting logical connection between these, and so we have split them in our formalisation into Axiom I, 3.1, I, 3.2, I, 4.1 and I, 4.2, giving a total of 10 axioms.

Axioms I, 1 and I, 2, which were a single axiom in the first edition of the *Grundlagen der Geometrie*, require that two points uniquely determine a line. Analogous axioms for planes are given by I, 4.1 and I, 5. The converse, namely that a line is determined by two points, appears with the addition of Axiom I, 3.1. The converse for planes, that a plane is determined by three non-collinear points, was an axiom of the first-edition, but was later weakened to assert only that a plane contains at least one point in Axiom I, 3.2. Hilbert must have been aware that the former could be derived from the latter when he removed the redundancy, but a statement of this fact and the proof are both absent from the text. We present our own proof and verification in §3.2.4.

We should note that there is some ambiguity in Axiom I, 3.1. Is Hilbert saying that there are two points and some line such that the points lie on that line, or is he saying more generally that *every* line contains two points? We took the latter view, since on the weaker interpretation, we could formalise a model of the first group of axioms following the technique in §6.3.2, and show in this model that there is a line with *no* incident points. We assume Hilbert did not intend this.

Next, we have Axiom I, 3.2. This is a *dimension* axiom, requiring that the geometry has at least dimension 2. An analogous axiom is the last axiom (I, 8), which requires that the geometry has at least dimension three. We will have very little need for this last axiom, since almost all of Hilbert's proofs are basically planar.

Finally, we have the Axioms I, 6 and I, 7. Together, these require that intersecting planes meet in a line, and thus, they restrict the dimension of the geometry to 3.

It is important to note that Hilbert adopts the uncommon convention that when he writes expressions such as "two points", "three points", or "two lines", "three lines", he is assuming that the points and lines in question are distinct. For this reason, a number of explicit distinctness assumptions appear in our formalisation which are only implicit in the prose. Some of these distinctness assumptions can actually be dropped,

$$\vdash A \neq B \implies \exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \tag{I, 1}$$

$$\begin{aligned} \vdash A \neq B &\wedge \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \\ &\wedge \texttt{on\_line}\ A\ b \wedge \texttt{on\_line}\ B\ b \\ &\implies a = b \end{aligned} \tag{I, 2}$$

$$\vdash \exists A.\ \exists B.\ A \neq B \wedge \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \tag{I, 3.1}$$

$$\vdash \exists A.\ \exists B.\ \exists C.\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \tag{I, 3.2}$$

$$\begin{aligned} \vdash \neg(\exists a.\ &\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\implies \exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \end{aligned} \tag{I, 4.1}$$

$$\vdash \exists A.\ \texttt{on\_plane}\ A\ \alpha \tag{I, 4.2}$$

$$\begin{aligned} \vdash \neg(\exists a.\ &\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\wedge \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \\ &\wedge \texttt{on\_plane}\ A\ \beta \wedge \texttt{on\_plane}\ B\ \beta \wedge \texttt{on\_plane}\ C\ \beta \\ &\implies \alpha = \beta \end{aligned} \tag{I, 5}$$

$$\begin{aligned} \vdash A \neq B &\wedge \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \\ &\wedge \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \\ &\implies \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha \end{aligned} \tag{I, 6}$$

$$\begin{aligned} \vdash \alpha \neq \beta \wedge \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ A\ \beta \\ \implies \exists B.\ A \neq B \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ B\ \beta \end{aligned} \tag{I, 7}$$

$$\begin{aligned} \vdash \exists A.\ \exists B.\ \exists C.\ \exists D.\ \\ \neg\exists \alpha.\ (\texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha) \end{aligned} \tag{I, 8}$$

Figure 3.2: Group I axioms

such as in Axiom I; there is a line through the points A and B whether or not A and B are distinct. We keep the weaker form as the axiom, and verify the stronger version.

$$\vdash \exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a.$$

3.2.3 Related Axiomatisations

Hilbert's axiomatisation appears within a culture of related attempts to rigorise geometry. Oswald Veblen's doctoral work [100] is perhaps the most closely related, and it is clear that ideas developed by Veblen and his supervisor E. H. Moore filtered into later editions of Hilbert's text.

Veblen differs significantly from Hilbert by following a trend he identifies with Pasch and Peano. Here, the fundamental primitives of geometry are just points and the relation of *betweenness*. Lines, planes and incidence are no longer primitive, but are instead derived concepts.

A similar approach was taken up by Tarksi, who developed the first formal system for elementary geometry with the benefit of modern formal logic [95]. Like Veblen, Tarski used only one primitive sort for points, but unlike Veblen, he admitted a congruence relation on pairs of points. Nevertheless, Tarski's axioms are particularly elegant. They do not appeal to complex derived notions as Hilbert's later axioms do, and his dimension axiom has the pleasing property that it can be mechanically modified to axiomatise an arbitrary dimension.

There are mechanisations of Tarski's geometry in the Otter automated prover [80] and Coq proof assistant [77]. Tarski's axioms are known to be far more primitive than Hilbert's, and thus it takes much more work to carry out even simple verifications in this system. In fact, it takes significant effort just to recover Hilbert's axioms from Tarski's [7]. That said, Tarski's theory embeds in the theory of real-closed fields where it admits quantifier elimination, and thus the theory is *decidable*. The axioms are still very *hard* to work with. In fact, the decision procedure is doubly-exponential! [17]

3.2.4 Elementary Consequences

Hilbert highlights two results for his first group.

> THEOREM 1. Two lines in a plane either have one point in common or none at all. Two planes have no point in common, or have one line and

> otherwise no other point in common. A plane and a line that does not lie in it either have one point in common or none at all. [1]
>
> THEOREM 2. Through a line and a point that does not lie on it, as well as through two distinct lines with one point in common, there always exists one and only one plane.
>
> [42, p. 4]

The proofs are straightforward. In fact, the first and last clauses in THEOREM 1 are barely rewordings of Axiom I, 2 and Axiom I, 6, and so we did not bother to formalise them.

The middle clause, that two planes have either no point in common or otherwise one line and no other point in common is a little more involved. We start by giving it a tidier rephrasing which makes it easier to formalise: two planes with a point in common intersect exactly in some line.

$$\vdash \alpha \neq \beta \wedge \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ P\ \beta \\ \Longrightarrow\ (\exists a.\ \forall Q.\ \texttt{on\_plane}\ Q\ \alpha \wedge \texttt{on\_plane}\ Q\ \beta \iff \texttt{on\_line}\ Q\ a).$$

The verification of this theorem, and the first piece of Mizar Light that we present, is given in Figure 3.3. We encourage the reader to inspect these verifications, and we hope they agree with us that they are pleasantly readable and easy to follow.

THEOREM 2, split into two separate propositions, is easily verified. The verifications depend on Axioms I, 2, I, 3.1, I, 4.1, I, 5 and I, 6.

$$\vdash \neg\texttt{on\_line}\ P\ a \Longrightarrow \exists!\alpha.\ \texttt{on\_plane}\ P\ \alpha \wedge \forall Q.\ \texttt{on\_line}\ Q\ a \Longrightarrow \texttt{on\_plane}\ Q\ \alpha.$$

$$\vdash a \neq b \wedge \texttt{on\_line}\ P\ a \wedge \texttt{on\_line}\ P\ b \\ \Longrightarrow\ \exists!\alpha.\ \forall P.\ \texttt{on\_line}\ P\ a \vee \texttt{on\_line}\ P\ b \Longrightarrow \texttt{on\_plane}\ Q\ \alpha.$$

Note that the symbol $\exists!$ is another defined quantifier in HOL Light. It asserts of its argument — a predicate — that it is satisfied *uniquely*, and it can be defined thus:

$$\vdash_{def}\ (\exists!P) = (\exists P) \wedge (\forall x.\ \forall y.\ P\ x \wedge P\ y \Longrightarrow x = y)$$

[1] We need classical logic already here. The very first clause of THEOREM 1 assumes that point equality is decidable. See Dehlinger et al [21].

```
assume α ≠ β ∧ on_plane P α ∧ on_plane P β                                    1
so consider Q such that P ≠ Q ∧ on_plane Q α ∧ on_plane Q β by (I, 7)          2
so consider a such that on_line P a ∧ on_line Q a by (I, 1)                   3
take a
fix R
have on_plane R α ∧ on_plane R β ⟹ on_line R a
proof
    assume on_plane R α ∧ on_plane R β                                        4
    otherwise assume ¬on_line R a                                             5
    hence ¬(∃a. on_line P a ∧ on_line Q a ∧ on_line R a) from 2,3 by (I, 2)
    qed from 1,2,4 by (I, 5)
qed from 1,2,3 by (I, 6)
```

Figure 3.3: Intersecting planes intersect in a line

We will not reproduce the verifications here, since our later verifications have no need of these theorems, and nor does Hilbert refer to THEOREM 2 again. Instead, all our verifications which appeal to incidence will be based on an alternative formulation and some formal theory which we explain in §3.2.6.1.

The only other theorem we need is one we use when developing our formal theory of half-planes in Chapter 7. It verifies that every plane contains a non-collinear triple, an axiom of the first edition but a result that now needs to be proven. We regard it as an oversight of Hilbert's that he neglected to mention the result, let alone give its proof. It is not entirely trivial, having us consider a configuration of six points and three planes. We give a prose proof now.

Proposition. *Every plane* α *contains at least three non-collinear points.*

Proof. By Axiom I, 4.2 and Axiom I, 8, we can take a point A on α and a point B not on α. We connect these two points by a line a. By Axiom I, 3.2, we can take a third point C off the line a. The three points A, B and C must determine a plane β by Axiom I, 4.1.

Since the planes α and β intersect at the point A, we can choose another intersection

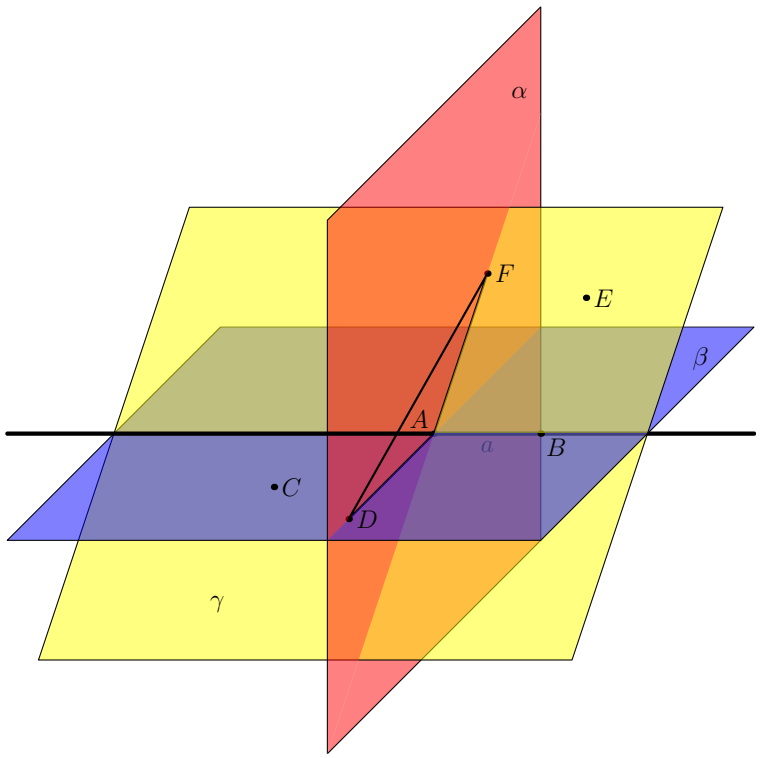


Figure 3.4: Planes contain non-collinear points

point D by Axiom I, 7, and by Axiom I, 8, we can find a fifth point E off the plane β. Now A, B and E must be non-collinear, and so they determine a plane γ. If this plane is α, then A, B and E are our three points and we are done. Otherwise, we have two distinct planes intersecting at A and so we can take another intersection point F by Axiom I, 7. This gives us three non-collinear points in α, namely A, D and F. See Figure 3.4.

$$\vdash \exists A.\ \exists B.\ \exists C.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \\ \wedge \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a). \quad (3.1)$$

□

This is the only three-dimensional theorem we will consider in the present work, and as such, it is the only theorem which depends on Axiom I, 8.

3.2.5 Absent Arguments

After splitting conjunctions, we found that there were ten axioms in Hilbert's first group. This is twice as many axioms as the next largest group, Group III. One would

expect, then, that these axioms would feature the most in proofs. This is indeed the case with formal verifications, but in Hilbert's prose, the axioms are almost never cited.

This appears to directly contradict Weyl, who claimed that the deductions in Hilbert's geometry contain no gaps [103]. Indeed, taking his claim at face-value, we would have to conclude with Meikle and Fleuriot [66] that Hilbert's arguments are full of missing assumptions and lemmas.

We do not believe Hilbert ever made this claim, given that he is happy to elide whole proofs. The only standard we hold Hilbert to is that his deductions are logical consequences of previous ones, and that he has made a reasonable balance in his presentation, carefully elucidating only the particularly tricky proofs. So far, we have confirmed that all of his deductions are indeed valid. It is less clear whether his presentation is balanced, as we shall discuss over the remaining chapters.

What is clear, though, is that there are many weaknesses in Hilbert's presentation when viewed as a guide for mechanical verification. Hilbert's decisions to cite axioms are sometimes erratic (see §5.3.3), and he includes proofs which are easily verified while omitting others he claims to be easily provable when they are a challenge to verify.

There is one axiom which Hilbert *is* careful to cite. This is Axiom I, 3 (or more precisely in our formalisation, Axiom I, 3.2). We might suppose Hilbert cites this axiom because it is usually used to introduce points. In an informal geometry proof, where the goal is to obtain a geometric figure, one would not want to leave this sort of introduction step implicit.

For many of the other axioms whose citations are missing, we might still excuse Hilbert by noting again that almost all of his proofs are *planar*. In effect, Hilbert makes a pervasive "without-loss-of-generality" assumption, which is justified because the axioms he invokes will always force the objects considered to lie in the same plane. This makes his life substantially easier, since it means he is effectively working with just Axioms I, 1, I, 2, I, 3.1 and I, 3.2.

It would have made our verification effort somewhat easier had we been able to make this same without-loss-of-generality assumption. A simple idea would have been to develop a purely planar theory of geometry which could then be embedded in the individual planes of a space geometry. Unfortunately, we knew of no way to do this using HOL Light's simple type theory. Theory embeddings which depend on particular planes would suggest we need at least a dependently typed logic such as Coq's [111].

3.2.6 Point sets

In our first attempts to verify Hilbert's first few groups, using just the stock automation available to our theorem prover, we found that many steps were needed that did not appear explicitly in the prose. These steps almost always concerned incidence, and were rarely enlightening. The fully explicated verifications were difficult to read, and could not be easily used to comment on Hilbert's presentation, to identify dependencies, redundancies, circularities, missing details, or alternative proof strategies.

Unsatisfied by this, we tried to improve the situation by reformulating statements of incidence in a way which was more expressive than the basic primitives.[2] Realising that the domain of incidence reasoning is inherently combinatorial, we opted to formulate incidence claims in terms of point sets. One advantage of point sets is that they at least have the possibility of being composed via basic operations on sets.

Thus, we defined two predicates:

$$\texttt{collinear} : (\texttt{point} \to \texttt{bool}) \to \texttt{bool}$$
$$\vdash_{def} \texttt{collinear Ps} \iff \exists a.\, \forall P.\, P \in Ps \implies \texttt{on\_line}\, P\, a.$$

$$\texttt{planar} : (\texttt{point} \to \texttt{bool}) \to \texttt{bool}$$
$$\vdash_{def} \texttt{planar Ps} \iff \exists \alpha.\, \forall P.\, P \in Ps \implies \texttt{on\_plane}\, P\, \alpha.$$

We now refer to lines and planes by using the points which uniquely identify them. So if two distinct points A and B lie on a line a, we are able to formalise a statement that two other points C and D also lie on a by writing

$$\texttt{collinear}\ \{A,B,C,D\}.$$

Furthermore, we can formalise a claim that another point E does *not* lie on the line a with $\neg\texttt{collinear}\ \{A,B,E\}$. Notice that we only need to use three points in this formula, and that adding more points only weakens it (every non-collinear set has a three-point non-collinear subset). This means that formulas asserting non-incidence now assert the existence of *triangles*.

The real advantage to be gained by using collinear and planar sets is that we were able to capture the logic of incidence reasoning in terms of individual composition theorems for sets. Our early investigations into Hilbert's verification [86] indicated that these theorems could take on the bulk of the incidence reasoning needed to verify Hilbert's proofs.

[2] Note that we do not change any axioms. We simply define our new formulation in terms of the old.

$$\vdash \texttt{collinear}\ \{A,B\} \tag{3.2}$$

$$\vdash S \subseteq T \wedge \texttt{collinear}\ T \implies \texttt{collinear}\ S \tag{3.3}$$

$$\vdash A \neq B \wedge A,B \in S,T \implies \texttt{collinear}\ S \wedge \texttt{collinear}\ T \implies \texttt{collinear}\ (S \cup T) \tag{3.4}$$

$$\vdash \texttt{planar}\ \{A,B,C\} \tag{3.5}$$

$$\vdash S \subseteq T \wedge \texttt{planar}\ T \implies \texttt{planar}\ S \tag{3.6}$$

$$\vdash \neg\texttt{collinear}\ (S \cap T) \wedge \texttt{planar}\ S \wedge \texttt{planar}\ T \implies \texttt{planar}\ (S \cup T) \tag{3.7}$$

$$\vdash \texttt{collinear}\ S \implies \texttt{planar}\ S \tag{3.8}$$

$$\vdash A \neq B \wedge A,B \in S,T \wedge \texttt{collinear}\ S \wedge \texttt{planar}\ T \implies \texttt{planar}\ (S \cup T) \tag{3.9}$$

$$\vdash P \in S \wedge P \in T \wedge \texttt{collinear}\ S \wedge \texttt{collinear}\ T \implies \texttt{planar}\ (S \cup T) \tag{3.10}$$

Figure 3.5: Derived incidence theorems in point sets

3.2.6.1 Incidence Reasoning with Point Sets

In Figure 3.5, we give a set of verified theorems for reasoning with collinear and planar sets. Note that we have assumed that singleton sets are both collinear and planar, and that the empty set is assumed to be the smallest collinear and planar set.

So long as we are restricting our attention to a finite number of points, which is the typical context for applying these theorems, we can see how they reflect those incidence axioms which do not introduce points, and thus justify them as an alternative way to understand the logic of incidence.

We want this important detail, since it is not our intention to introduce a bunch of *ad hoc* theorems which happen to work most of the time. We want a genuinely alternative way to think about incidence reasoning, one which has nice computational properties.

We explain the content of our theorems thusly: given a finite set of points S, let us think of a line containing the points of S, should it exist, as a maximal collinear superset of S. Similarly, let us think of a plane containing the points of S, should it exist, as a maximal planar superset of S. In this way, we can define lines and planes entirely in terms of point-sets known to be collinear and planar.

In this sense, Theorem 3.2 must assert that the points A and B are incident with a line AB and corresponds to Axiom I, 1. Theorem 3.4 effectively asserts that the expression

"the line *AB*" is well-defined, and thus corresponds to Axiom I, 2. To see this, recall that the line *AB* is the unique maximal superset of all points containing *A* and *B*. Now the largest possible set containing *A* and *B* is just the finite union of all sets containing *A* and *B*. What Theorem 3.4 is then telling us is that this set is collinear. In other words, the unique largest set containing *A* and *B* is the line of *AB*.

Similarly, Theorem 3.5 tells us that *A*, *B* and *C* are incident with the plane *ABC* while Theorem 3.7 asserts that the expression "the plane *ABC*" is well-defined, or more generally, that a plane is uniquely determined by any of its non-collinear subsets.

Theorem 3.9 is a stronger version of Axiom I, 6. To see this, we again think of our lines and planes as maximal collinear and planar sets. Axiom I, 6 says "[i]f two points *A*, *B* of a line *a* lie in a plane α then every point of *a* lies in the plane α." Here, we take the line *a* to be a maximal collinear set *S* and α to be a maximal planar set *T*. According to Theorem 3.9, $S \cup T$ is also planar. But *T* is maximal, so we must have $S \cup T = T$ and thus $S \subseteq T$. In other words, all points of the line *S* lie in the plane *T*.

Finally, Theorem 3.10 tells us that distinct intersecting lines lie in a unique plane, the claim made in Hilbert's THEOREM 2. This is because their union must be non-collinear, since otherwise they would not be maximal sets. Thus, their union determines a plane.

The other theorem, THEOREM 1, notes firstly that distinct lines have either no points in common or just one point in common. This follows directly from Theorem 3.4. Indeed, distinct lines as maximal collinear sets must have a non-collinear union, so they cannot have more than one point in common.

THEOREM 1 further notes that distinct planes have either no points in common or one line in common. We know that distinct planes as maximal planar sets must have non-planar unions and so must have collinear intersections by Theorem 3.7. Moreover, we know that if the intersection contains two points, then, according to Theorems 3.4 and 3.9, the two points yield a maximal collinear set contained in the maximal planar set. We will not say anything about the case of a one-point intersection, since this requires the existential Axiom I, 7: if two planes have one point in common, they have at least one other point in common. As mentioned above, we do not consider such point introduction axioms here.

In conclusion, we have shown how all incidence axioms which do not introduce points can be expressed and strengthened as composition theorems on collinear and planar sets. Primitive lines and planes are no longer used directly, since all the axioms gov-

erning them can be subsumed by these composition theorems.

3.2.6.2 Evaluation

Meikle's verifications of Hilbert's geometry involved a lot of tedious but necessary reasoning about incidence relations, and the derivation of many additional lemmas to support the main verifications. With Theorems 3.2–3.10, the verifications are a good deal less complex. For instance, the verification of THEOREM 3, which relies heavily on incidence reasoning, needed twenty-seven special case lemmas and forty steps in Meikle's verification, while our own verification using the above theorems had twenty-two steps and no additional lemmas.

This was an improvement, but it still left our verifications bogged down in trivial combinatorial details that made them difficult to compare to the prose. Besides, the verifications were still very difficult to obtain, since we almost always needed to figure out the specific point sets with which to manually specialise the quantifiers in Theorems 3.2–3.10: the proof assistant could not figure these out for itself with generic automation. Finding these sets is not just tedious, but error-prone. When our verifications were correct, they were often suboptimal. And in one case, our difficulty in proving certain properties of a geometric configuration led us to believe, mistakenly, that Hilbert had made an error in one of his arguments (see §5.3.1.1).

Luckily, when we reflected on our manual verifications, we realised that Theorems 3.2–3.10 were always applied systematically. In the next chapter, we shall describe how to make these theorems the basis for an automated tool which can completely hide the messy incidence reasoning. We can then justify Hilbert's omission of the incidence arguments by claiming that they consist merely of exhaustive combinatorial reasoning which requires no geometric insight. In a geometric proof, the important steps are those which introduce points and thereby build up the geometric configuration. Hilbert's prose proofs, and our own verifications backed up by our incidence automation, leave just those steps explicit.

We can then continue to view Hilbert's proofs as *model proofs* and argue that, with our automation handling the tedium of combining point sets, our verifications will be good substitutes for their missing prose counterparts. They will be more trustworthy whilst being close to what a working mathematician would produce.

We finish by remarking that Theorems 3.2–3.10 were verified in a procedural rather than declarative style in HOL Light. These theorems will become the implementation

details of an incidence reasoning algorithm, rather than theorems of general geometric interest. It therefore made sense to take full advantage of HOL Light's tactics and its simplifier, which are particularly effective when working with finite sets.

3.3 Group II

With Hilbert's second group of axioms, we have our ordered geometry, which delimits the scope for the present work. We only have a few geometrical notions to hand, and things might seem quite restrictive, but we have enough to verify our main result: the Polygonal Jordan Curve Theorem.

Order axioms were missing in the ancient axiomatisations of geometry such as Euclid's, and their introduction marks an important milestone in the modern rigorisation of geometry. The first investigation of order axioms is credited to Pasch [79], and to this day, Hilbert's one planar axiom in this group and its variants elsewhere are still referred to as *Pasch's Axioms.*

Like the first group, Hilbert's second group of axioms went through substantial revision between editions. There was initially a great deal of redundancy, the investigation of which was made by other contributors. Huntingdon and Kline gave a thorough analysis of axioms for ordering along a line [45], while E.H. Moore and his student Veblen showed how, via Pasch's Axioms, Hilbert's main linear axiom was derivable. Veblen showed great interest in a bare ordered geometry, proving forty results compared to Hilbert's ten. He gave an early proof attempt of the Polygonal Jordan Curve Theorem (see Chapter 8), and later set out to recover the full metrical Euclidean geometry using only order axioms, the parallel axiom and a continuity axiom.

3.3.1 Axioms and Primitive Notions

Hilbert's second group supplies a single new primitive, namely *betweenness*, with which one can form expressions such as "the point *B* lies *between A* and *C*". We are supposed to interpret these expressions strictly, as required by Hilbert's first axiom:

> II, 1 *If a point B lies between a point A and a point C then the points A, B, C are three distinct points of a line, and B then also lies between C and A.*
>
> [42, p. 5]

The fact that Hilbert needs to axiomatically assert the irrelevance of the order of A and C tells us that we are formally working with a three place relation:

$$\texttt{between} : \texttt{point} \to \texttt{point} \to \texttt{point} \to \texttt{bool}.$$

This first axiom is not particularly informative. It really just gives some useful conditions on the betweenness relation. The symmetry requirement could have been dropped had we instead used a predicate of type $\texttt{point} \to \texttt{pair point} \to \texttt{bool}$ where `pair` is the type constructor for unordered pairs. We could have dropped the strictness requirement that all points are distinct as Tarski does in his axiomatisation, and we could have allowed degenerate betweenness assertions when the three points are non-collinear. The only consequence would be that other axioms and theorems would sometimes have to make additional non-degeneracy assumptions.

We have formalised a weaker version of this axiom than the one given by Hilbert. With the other order axioms, the stronger version is verifiable. The weakening arises because we do not conclude that all points are distinct, only that A is distinct from C.

$$\begin{aligned} \vdash \texttt{between}\ A\ B\ C \implies & A \neq C \\ & \wedge (\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ & \wedge \texttt{between}\ C\ B\ A. \end{aligned} \tag{II, 1}$$

We now give the remaining axioms.

> II, 2 *For two points A and C, there always exists at least one point B on the line AC such that C lies between A and B.*
>
> II, 3 *Of any three points on a line there exists no more than one that lies between the other two.*
>
> II, 4 *Let A, B, C be three points that do not lie on a line and let a be a line in the plane ABC which does not meet any of the points A, B, C. If the line a passes through a point of the segment AB, it also passes through a point of the segment AC, or through a point of the segment BC.*
>
> [42, p. 5]

Axiom II, 2 can be compared to Euclid's second postulate: "To produce a finite straight line continuously in a straight line." Here, Hilbert is telling us that we can extend the segment AC to a point B in the direction $\overrightarrow{AC}$. This axiom is absolutely key in building up geometrical figures.

$$\vdash \texttt{between}\ A\ B\ C \implies A \neq C$$
$$\wedge\ (\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \quad \text{(II, 1)}$$
$$\wedge\ \texttt{between}\ C\ B\ A$$

$$\vdash A \neq B \implies \exists C.\ \texttt{between}\ A\ B\ C \quad \text{(II, 2)}$$

$$\vdash \texttt{between}\ A\ B\ C \implies \neg\texttt{between}\ A\ C\ B \quad \text{(II, 3)}$$

Pasch's Axiom
$$\vdash \neg(\exists b.\ \texttt{on\_line}\ A\ b \wedge \texttt{on\_line}\ B\ b \wedge \texttt{on\_line}\ C\ b)$$
$$\wedge\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha$$
$$\wedge\ (\forall P.\ \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha)$$
$$\wedge\ \neg\texttt{on\_line}\ A\ a \wedge \neg\texttt{on\_line}\ B\ a \wedge \neg\texttt{on\_line}\ C\ a$$
$$\wedge\ \texttt{on\_line}\ P\ a \wedge \texttt{between}\ A\ P\ B$$
$$\implies \exists Q.\ \texttt{on\_line}\ Q\ a \wedge (\texttt{between}\ A\ Q\ C \vee \texttt{between}\ B\ Q\ C)$$
(II, 4)

Figure 3.6: Group II axioms

Axiom II, 3 is akin to an anti-symmetry property for linear ordering. With Axiom II, 1, it also shows that from `between` A B C we can infer that A, B and C are mutually distinct.

Axiom II, 4 is Pasch's axiom. So far, it is the most complex axiom we have to apply, being a planar axiom with numerous preconditions and a disjunctive conclusion wrapped in an existential. The complexity can be measured by comparing the size of the formalisation with that for the other axioms. See Figure 3.6.

3.3.2 Pasch and Incidence Reasoning

Our early experiences verifying Hilbert's early theorems showed that most of the effort is expended trying to verify the preconditions of Pasch's Axiom (II, 4). In order to leverage our representation in point sets, we decided to derive another formalisation of the axiom in terms of collinearity and planarity. To do this, we remove all mention of the line a from the axiom, and replace it by two defining points D and E. The point D will be assumed to be the point of intersection between the line a and the segment AB, and the point E will be any other point on the line a. See Figure 3.7.

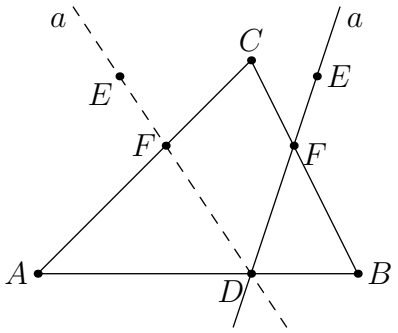


Figure 3.7: Axiom II, 4

Our preconditions can now be expressed in terms of non-collinear and planar sets, as seen in the following verified formulation of Axiom II, 4:

$$\begin{aligned}\vdash \neg\texttt{collinear}\ \{A,B,C\} \wedge \neg\texttt{collinear}\ \{A,D,E\} \wedge \neg\texttt{collinear}\ \{C,D,E\} \\ \wedge \texttt{planar}\ \{A,B,C,D,E\} \wedge \texttt{between}\ A\ D\ B \\ \Longrightarrow \exists F.\ \texttt{collinear}\ \{D,E,F\} \wedge (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C).\end{aligned} \tag{3.11}$$

Looking at this theorem, it is hopefully clearer how the theorems from Figure 3.5 are needed when reasoning about incidence in Hilbert's proofs. Most of his results in Group II require reasoning about order in the plane by applying Pasch's Axiom. Each time the axiom is applied, we must verify the preconditions of the axiom, which according to our formalised Theorem 3.11, means we must find three triangles and a planar set.

This is not all. Typically, we must also eliminate one of the disjuncts in the conclusion of the axiom, and this requires further incidence reasoning. Usually, we show how, in one branch, all points considered end up collapsing to just a single line. This will contradict our assumptions that we have at least one triangle.

Our verifications from our earlier work were laden down with steps to find triangles by repeatedly applying the theorems from §3.2.6.1. We will show some of the complexity in Chapter 5, but will see how, luckily, it can be fully automated.

3.4 Conclusion

The formalisation of Hilbert's first group of axioms and the verification of his first two theorems is straightforward, up to a few minor technical points about the choice of representation and whether we implicitly assume that the primitive sorts or types are in-

habited. Otherwise, the axioms of Group I are conspicuous only for their absence from Hilbert's proofs, especially in Group II where incidence reasoning is needed heavily in order to apply Pasch's axiom. However we are to justify the absence, all details must be restored in our verifications. When we do so, we find our verifications are washed out with fussy incidence arguments.

By reformulating in terms of point sets, we can alleviate this somewhat, but it only takes us so far. If we are to keep our proofs as clean as Hilbert's, and have a decent chance of verifying more complex theorems, we will want to make almost all the incidence reasoning implicit. This we leave to the next chapter, where we consider automation.

Chapter 4

Automation

In the last chapter, we formulated incidence claims in terms of point-sets, characterised by Theorems 3.2–3.10 from Figure 3.5. These allowed us to write much shorter verifications [86] than had been obtained by Meikle and Fleuriot [66], just using the theorem prover's stock automation. But we knew we could do better. The patterns that appeared in these verifications, which we shall refer to as our *manual* verifications, were ripe for additional automation.

The HOL Light proof assistant expects its users to work "close to the metal", writing ML directly. In this environment, users become comfortable prototyping and integrating new tools. In this chapter, we shall describe the integration of an automated tool for incidence reasoning, which can be made available as a procedural tactic and to Mizar Light proof steps.

Much of this chapter and the subsequent chapter is an expansion and improvement on earlier work [87, 88, 89]. In it, we shall show how the user and tool can assist each other concurrently, thereby collaborating as the user develops a proof.

4.1 Background

So far as fully automated theorem proving goes, the oldest successes are probably in geometry. Unfortunately, the approaches used assume much more than the very general theory of incidence that we consider here, and so we have had to develop our own methods. We just briefly mention some others.

4.1.1 Wu's Method

When considering general automation in Hilbert's *Grundlagen der Geometrie*, there is probably no work more relevant than Wu's method [96]. Wu credited the *Grundlagen* for the metatheoretical insights that led to his mechanisation procedure for the whole of geometry.

One of those insights was that the method of proof in a synthetic geometry system such as Hilbert's often falls short of absolute rigour because degenerate cases are routinely missed. Typically, when we state axioms and properties of some geometric figure, we have in mind a "genericity" of that figure which is hard to capture formally. Our statements may admit some generalisation to what we would regard as degenerate cases, even if this is not immediately clear at the time. We have already seen that some of Hilbert's axioms, such as Axiom I, 1, when combined with other axioms, generalise to degenerate cases, and we shall see this again with Axiom II, 4, which we shall strengthen in Chapter 7.

But it is not always clear how the conditions in our statements can be relaxed. Moreover, when trying to rule out certain degenerate cases, there are plenty that are easily missed. When we formalise, we cannot simply neglect them unless we have proof tools that can do it for us. Filling in all the gaps requires enormous effort and complication of the verification, so it would seem that Wu is correct: we cannot be truly rigorous unless we can systematically deal with degenerate cases. This provides an alternative way to diagnose the gaps in Hilbert's proofs spotted by Meikle and Fleuriot [66]: they were gaps concerning degenerate cases of point incidence.

Wu's highly celebrated method automatically inserts non-degeneracy conditions as it carries out proofs, and if possible, it automatically deletes redundant conditions to keep results generic. The method has been used to automatically prove an enormous number of non-trivial results in unordered geometry [90].

For ordered geometry, Wu extended his method by appealing to the embedding of Euclidean geometry in real-closed fields, and then applied Tarksi's well-known decision procedure. Unfortunately, the procedure is grossly intractable [64]. Thus, in the domain of ordered geometry, which is our principle concern, Wu's Method will not be effective.

Moreover, to exploit Wu's method, we would need to show how each of his mechanical steps can be reduced to the axioms of Hilbert's system. But this reduction will presuppose some of the very elementary results which we are trying to mechanise.

Indeed, Wu's method rests at least on Desargues' Theorem, which appears late on as THEOREM 53 (page 72 of the *Foundations of Geometry*).

4.1.2 Other Methods

Wu's technique is algebraic, and reduces geometrical problems to polynomial equations. A similar successful method was developed as *Gröbner bases* [8]. Like Wu's, this method automatically infers non-degeneracy conditions (though will not automatically delete redundant ones). The two techniques are comparable in the results they obtain, but in some cases, a problem soluble by one method may only be soluble by the other after some additional factorisation of the polynomial equations.

Another collection of successful methods which combine algebraic and synthetic reasoning can be described as *coordinate-free* techniques [91]. With these techniques, it is possible to eliminate many of the non-degeneracy conditions by using generic analogues of standard geometric properties. For instance, in the area method, we generalise the notion of the area of a polygon to a *signed* area according to its orientation, and in the full-angle method, we generalise the notion of an angle to an arbitrary pair of lines. Using a simple set of algebraic laws on signed areas and full-angles, it is possible to prove many geometric results *and* to preserve the geometric intuition behind them. The polynomial approaches, by contrast, will produce very dense systems of equations involving dozens of variables at which point the geometric intuition is lost.

The area method has been formalised as a Coq tactic by Narboux [76], while the full-angle method was used for Wilson and Fleuriot's diagrammatic geometry proof assistant [108].

4.2 Basis for an Algorithm

Our own approach is based on Theorems 3.2–3.10, which tell us how to reason with finite collinear and planar sets, and so can form the basis of a combinatorial algorithm. The domain of our algorithm is data representing assertions, hereafter just *data*.[1] There are five kinds of data: assertions that two points are equal, that two points are distinct, that a set of points is collinear, that a triple is non-collinear, and that a set of points is

[1] In our implementation, the data were precisely the sequents whose right hand side formalised the assertion, making our implementation *fully expansive* [5], but we could have used other data, such as a collection of proof hints, which could later be automatically processed into sequents.

planar. The algorithm consists chiefly of rules and methods to generate data from other data.

These rules are similar in spirit to those given by Magaud et al [60]. Their rules are also based on point-sets, but the authors have beautifully abstracted the core idea. In their paper, an n-dimensional set is assigned a *rank* of $n+1$. There are then key rules which assert that, given point sets X and Y, the sum of the ranks of $X \cup Y$ and $X \cap Y$ is no greater than the sum of the ranks of X and Y. This abstractly characterises Theorems 3.4 and 3.7 which introduce the union of collinear and planar sets respectively, while the hierarchy of ranks give us Theorems 3.3 and 3.6 for free. The use of ranks therefore identifies an analogy between how collinearity and planarity are used in our theorems.

The approach taken by Magaud et al allows them to generalise to arbitrary dimension, and the elegance of the theory helps us see our own approach as less *ad hoc* than it might otherwise. For the rest of this chapter, however, we shall focus on our original, more concrete representation.

4.2.1 Inference Rules

Theorems 3.2–3.10 already show how to introduce collinear and planar sets. Additionally, we need ways to derive the inequalities and triangles which are conditions of the theorems. Firstly, we note that any triangle or non-collinear triple implies the mutual distinctness of its three points, giving us one way to introduce point inequalities. Another way to derive inequalities is through the following simple argument: suppose we have a collinear set S, and a non-collinear triple sharing two points with S. Then the third point of the triple must be distinct from all points in S.

Next, we need to introduce non-collinear triples. We based our method here directly on a common pattern of reasoning from our manual verifications. Suppose we have a collinear set S and a non-collinear triple sharing two points with S. Then the third point forms a non-collinear triple with all pairs of points in S known to be distinct.

Finally, we consider how we might infer when two points are equal using our theorems. Given two collinear sets S and T, which have a non-collinear union, we can infer that their intersection must be empty or a singleton. So if the intersection is a set of points $\{P_1, P_2, \ldots, P_n\}$, then we know immediately that these points are identical. This, we noted in the last chapter, is just telling us that distinct lines intersect in at most one

point, as per Hilbert's THEOREM 1.

We now summarise the rules for introducing our five kinds of data. The first four rules apply Theorems 3.2–3.10 indirectly. The last five are direct.

`ncolneq` Infer inequalities from non-collinear triples:

$$\vdash \neg\texttt{collinear}\ \{A,B,C\} \implies A \neq B \wedge A \neq C \wedge B \neq C$$

(by Theorem 3.2).

`colncolneq` Infer inequalities from a collinear set containing two points of a non-collinear triple:

$$\vdash \texttt{collinear}\ S \wedge \neg\texttt{collinear}\ \{A,B,C\}$$
$$\wedge A,B \in S \implies \forall X.\, X \in S \implies C \neq X$$

(by Theorem 3.3). For example,

$$\texttt{collinear}\{A,B,C,D,E\} \wedge \neg\texttt{collinear}\{A,B,P\}$$
$$\implies A \neq P \wedge B \neq P \wedge C \neq P \wedge D \neq P \wedge E \neq P.$$

`coleq` Equate points in the intersection of two collinear sets which are jointly non-collinear:

$$\vdash \texttt{collinear}\ S \wedge \texttt{collinear}\ T \wedge \neg\texttt{collinear}\ U \wedge U \subseteq S \cup T$$
$$\wedge A,B \in S,T \implies A = B$$

(by Theorems 3.3 and 3.4). For example,

$$\texttt{collinear}\{A,B,C,D,E\} \wedge \texttt{collinear}\{A,C,E,X,Y\}$$
$$\wedge \neg\texttt{collinear}\{A,B,Y\} \implies A = C \wedge A = E \wedge C = E.$$

`colncolncol` Infer new non-collinear triples from a collinear set and another non-collinear triple:

$$\vdash \texttt{collinear}\ S \wedge \neg\texttt{collinear}\ \{A,B,C\}$$
$$\wedge X,Y,A,B \in S \wedge X \neq Y \implies \neg\texttt{collinear}\ \{C,X,Y\}$$

(by Theorems 3.3 and 3.4). For example,

$$A \neq C \wedge D \neq E \wedge \texttt{collinear}\{A,B,C,D,E\} \wedge \neg\texttt{collinear}\{A,B,P\}$$
$$\implies \neg\texttt{collinear}\{A,C,P\} \wedge \neg\texttt{collinear}\{D,E,P\}.$$

colcol Use Theorem 3.4 to show that the union of collinear sets which intersect at more than one point is collinear.

planeplane Use Theorem 3.7 to show that the union of planar sets intersecting at a non-collinear triple is planar.

colplane Use Theorem 3.8 to show that a collinear set is planar.

colplaneplane Use Theorem 3.9 to show that the union of a collinear and planar set intersecting in at least two points is planar.

colcolplane Use Theorem 3.10 to show that the union of intersecting collinear sets is planar.

4.3 Forward Chaining

Forward-chaining algorithms have had recent success in automatically producing readable elementary proofs from Hilbert's axioms [85], and seemed particularly suitable for our use case. For instance, we found that when writing our manual verifications, we ended up with suboptimal proofs and found that the complexities of incidence reasoning left us with the suspicion that there might be errors in the prose where Hilbert had incorrectly assumed non-degeneracy conditions. We wanted a tool which could investigate these matters by exploring the proof space of incidence reasoning surrounding each of Hilbert's proofs.

The idea of using forward-chaining in this sort of exploratory way also opened up the possibility of designing an automated tool which could collaborate with the user as they develop a verification. Forward-chaining seemed quite apt, since its focus on cumulatively growing consequences of a set of assumptions corresponds well with how a declarative verification specifies a continually growing proof context.

Our basic approach is to produce cumulative *generations* of data by combining data from previous generations. For incidence reasoning, we shall be combining the data based on the procedures of §4.2.1, in the manner of forward-chaining.

4.3.1 Concurrency

Typically, the automation available in interactive proof assistants is invoked on demand by the user when they evaluate individual proof steps. But when the user writes the

formal proof for the first time, or comes to edit it later, they will spend most of their time thinking, consulting texts, backtracking and typing in individual proof commands. The CPU is mostly idling during this process, and we can exploit this idle time to run automated tools concurrently.

The Isabelle proof assistant has capitalised on this with Sledgehammer [68]. By invoking this command, the user can continue to work on a proof, while generic first-order tools are fired off as separate background processes, attempting to solve or refute the user's goals independently. If the tools reach a conclusion, the generated proof certificates can be automatically and seamlessly integrated into the user's proof-script.

We argue that we can do one better. We will show in §4.7.1 how to make a forward-chaining algorithm part of a "collaborative" architecture. As we remarked, both declarative proof and forward-chaining share the property of growing a proof context cumulatively, so why not splice the two? The user still manually crafts a search through the proof space, but they now have access to data from a forward-chaining algorithm. At the same time, the forward-chaining algorithm works independently, but can freely incorporate the user's intermediate hypotheses as they appear in the proof context, using them for its own derivations. The two systems, the automation and the human user, can thus be seen as collaborating, sharing data as they both strive towards the goal.

4.3.2 Discovery

In our designs, our automation is very much intended in the spirit of *assistance*. It is not intended to take over, or to solve the user's problems. It is there as a support, to be relied on as desired. Its purposes are thus quite modest, and it may not be clear why such a tool would be so desirable. Perhaps it would be useful to set the scene.

We would typically develop our geometric proofs on paper, figuring out how to discharge non-degeneracy assumptions without any computer assistance. In our case, these assumptions required that certain points be non-equal and others non-collinear. Discharging these assumptions typically meant a pen-and-paper combinatorial search.

In this narrow domain, it was easy to imagine simple programs which could do the combinatorial search for us, and which could display a breadth of derivations in case we had missed shorter paths, or overlaps with later inferences. This, then, is not quite automated search, but more assisted search. We have chosen to give it the shorter title of *discovery*, which stresses the point that we are usually interested in obtaining many

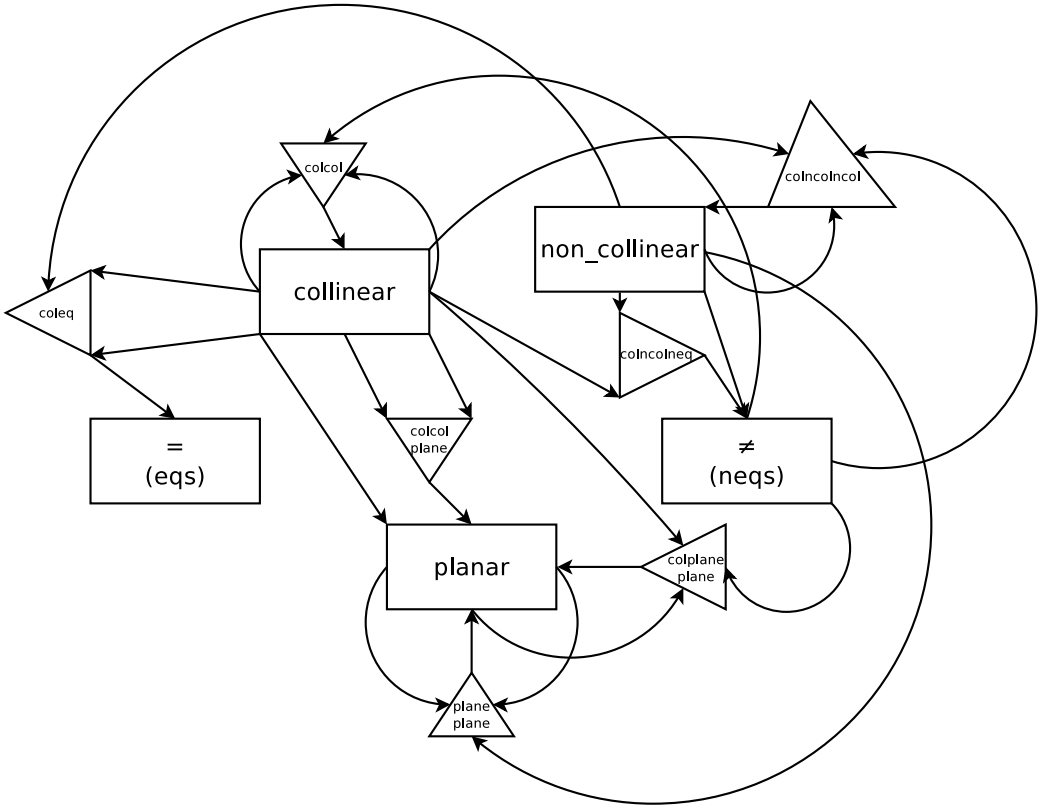


Figure 4.1: Data-flow for incidence reasoning. The five boxes represent our five lists of data, while triangles represent inference rules.

derivations from the search process, including derivations we had not expected. It also stresses the point that we often did not run the searches with a concrete goal in mind. However, we should mention that our use of the word *discovery* is to be contrasted with fully automated discovery systems which search for arbitrary interesting results and concepts without the need for assistance (see the work of Colton et al. [13, 14], McCasland and Bundy [63], Johansson et al [47] and Montano-Rivas et al [72, 73]).

4.4 An Implementation in Combinators

Our domain of incidence reasoning is naturally partitioned into data of five kinds. We aimed for an implementation which respected this partitioning, thereby greatly reducing the number of combinations of data we need to try with each rule from §4.2.1. These rules become the plumbing connecting the data. The data-flow model implied is given in Figure 4.1.

Our aim in this section will be to show how such data-flows can be expressed declara-

tively in a suitable combinator language. Combinator languages are a staple of typed functional programming and LCF theorem provers, which already boast powerful combinators for conversions and tactics [70]. By following the example of such languages, we hope to contribute a library that has a potentially wider application than the peculiar incidence automation we need for our verifications. The incidence automation will be just one possible hand-crafted *discoverer* written in this language, just as hand-crafted tactics are written in tactic languages.

The advantage of choosing a combinator language, as opposed to some other kind of domain-specific language, is that it fully integrates with the host programming language, and is therefore easy to integrate with the tactic combinator language and the Mizar Light combinator language. The user is also free to extend the language by defining derived combinators, and can easily inject their own computations into the language using lambda abstraction.

4.4.1 Related Work

The idea of an algebraic data-flow language was considered early on by Chen [10], who gave a specification for a variety of primitives very similar to our own, though without an implementation. Since then, algebras for logic programming, handling unbounded and fair search have been developed [54, 110], but these lack strong guarantees on the order in which elements are found, and can be unpredictable.

An equivalent version of the algebra we shall consider here was originally conceived of by Spivey [93]. It has been used recently by Katayama [52] to perform exhaustive searches of functional programs, and its theoretical underpinnings have been rigorously developed [92]. Unlike the other logic programming monads, it has much stronger constraints on the order in which values are searched for.

Our own algebra generalises Spivey's implementation somewhat by replacing his *bags* with more general collections, and we take advantage of this generalisation in §4.5.1. Moreover, our own realisation of this algebra offers a more "operational" motivation of the definitions to complement Spivey's.

4.4.2 Streams

Our overarching purpose is to output data, perhaps to a terminal, to a database to be used during a proof, or perhaps to another consumer for further processing. If we think

$$\begin{aligned}
&\texttt{fmap}\ (\lambda x.\ x)\ m = m \\
&\texttt{fmap}\ f \circ \texttt{fmap}\ g = \texttt{fmap}\ (f \circ g) \\
&\texttt{fmap}\ f \circ \texttt{return} = \texttt{return} \circ f \\
&\texttt{fmap}\ f \circ \texttt{join} = \texttt{join} \circ \texttt{fmap}\ (\texttt{fmap}\ f) \\
&(\texttt{join} \circ \texttt{return})\ m = m \\
&(\texttt{join} \circ \texttt{fmap return})\ m = m \\
&\texttt{join} \circ \texttt{join} = \texttt{join} \circ \texttt{fmap join}
\end{aligned} \tag{4.1}$$

Figure 4.2: The monad laws

of this output *as* the implementation, then we are dealing with procedures which lazily generate successive elements of a list. For the purposes of the theory, we assume that the lazy lists are infinite *streams*. These shall be the primitives of our data-flow algebra.

For now, we leave unspecified what computations are used to generate the primitive streams. It might be that a stream simply echoes a list of precomputed data; it might generate data based on standard input; it might generate them from some other automated tool. We shall focus instead on transformations for streams, and in how we might lift the typical symbolic manipulation used in theorem proving to the level of streams.

One reason why streams and lazy lists are a good choice here is that they generalise *the* ubiquitous data-structure of ML and its dialects, and have rich interfaces to manipulate them. A second reason why they are an obvious choice is that they have long been known to satisfy a simple set of algebraic identities and thus to constitute a monad [102]. We can interpret this monad as decorating computations with nondeterministic choice and backtracking search.

Monads themselves have become a popular and well-understood abstraction in functional programming. Formally, a monad is a type constructor M and three operations

$$\begin{aligned}
&\texttt{return} : \alpha \to M\ \alpha \\
&\texttt{fmap} : (\alpha \to \beta) \to M\ \alpha \to M\ \beta \\
&\texttt{join} : M\ (M\ \alpha) \to M\ \alpha
\end{aligned}$$

satisfying the algebraic laws given in Figure 4.2.

$$
\begin{array}{lccccccc}
 & [[D_{0,0}, & D_{0,1}, & D_{0,2}, & \ldots, & D_{0,n}, & \ldots], & \\
 & [[D_{1,0}, & D_{1,1}, & D_{1,2}, & \ldots, & D_{1,n}, & \ldots], & \\
 & [[D_{2,0}, & D_{2,1}, & D_{2,2}, & \ldots, & D_{2,n}, & \ldots], & \\
\text{shift} & [[D_{3,0}, & D_{3,1}, & D_{3,2}, & \ldots, & D_{3,n}, & \ldots], & \\
 & [[D_{4,0}, & D_{4,1}, & D_{4,2}, & \ldots, & D_{4,n}, & \ldots], & \\
 & [[D_{5,0}, & D_{5,1}, & D_{5,2}, & \ldots, & D_{5,n}, & \ldots], & \\
 & & & & & & \vdots &
\end{array}
$$

$$
\begin{array}{lccccccccc}
 & [[D_{0,0}, & D_{0,1}, & D_{0,2}, & D_{0,3}, & D_{0,4}, & D_{0,5}, & D_{0,6}, & D_{0,7}, & D_{0,8}, & \ldots], \\
 & & [D_{1,0}, & D_{1,1}, & D_{1,2}, & D_{1,3}, & D_{1,4}, & D_{1,5}, & D_{1,6}, & D_{1,7}, & \ldots], \\
 & & & [D_{2,0}, & D_{2,1}, & D_{2,2}, & D_{2,3}, & D_{2,4}, & D_{2,5}, & D_{2,6}, & \ldots], \\
 & & & & [D_{3,0}, & D_{3,1}, & D_{3,2}, & D_{3,3}, & D_{3,4}, & D_{3,5}, & \ldots], \\
= & & & & & [D_{4,0}, & D_{4,1}, & D_{4,2}, & D_{4,3}, & D_{4,4}, & \ldots], \\
 & & & & & & [D_{5,0}, & D_{5,1}, & D_{5,2}, & D_{5,3}, & \ldots], \\
 & & & & & & & [D_{6,0}, & D_{6,1}, & D_{6,2}, & \ldots] \\
 & & & & & & & & & & \vdots
\end{array}
$$

Figure 4.3: Shifting

The monad that is typically defined on lists can be used for search, but it takes concatenation `concat` : $[[\alpha]] \to [\alpha]$ for its `join` operation. This makes it unsuitable for *fair* and *unbounded search* with infinite streams. If the stream *xs* represents one unbounded search, then we have $xs + ys = xs$ for any *ys*, and thus, all items found by *ys* are lost.[2] This is to be expected, since the definition of this standard monad corresponds to *depth-first search*.

4.4.3 A Monad for Breadth-First Search

There is an alternative definition of the monad which is breadth-first and thus handles unbounded search. Here, the `join` function takes an infinite stream of infinite streams, and produces an exhaustive enumeration of all elements. We show how to achieve this in Figure 4.3 using a function `shift`, which moves each stream one to the "right" of its predecessor. We can then exhaustively enumerate every element, by enumerating each column, one-by-one, from left-to-right.

If we understand these streams as the outputs of a discoverer, then the outer stream

[2] Here, $+$ is just list and stream append.

can be understood as the output of a discoverer which *discovers discoverers*. The `join` function can then be interpreted as *forking* each discoverer at the point of its creation and combining the data into the output of a single discoverer. The highlighted column in Figure 4.3 is this combined result: a set of values generated simultaneously and thus having no specified order (this is required to satisfy Law 4.1 in Figure 4.2).

However, this complicates our stream type, since we now need additional inner structure to store the combined values. We will refer to each instance of this inner structure as a *generation*, following our terminology from §4.3. Each generation here is a finite collection of simultaneous values discovered at the same level in a breadth-first search. We just need to define the `join` function, taking care of this additional structure.

Suppose that generations have type $G\,\alpha$ where α is the element type. The manner in which we will define our `shift` and `join` functions on streams of generations assumes certain algebraic laws on the generations: firstly, they must constitute a monad; secondly, they must support a sum operation $(+) : G\,\alpha \to G\,\alpha \to G\,\alpha$ with identity $0 : G\,\alpha$.

The `join` function for streams must then have type $[G\,[G\,\alpha]] \to [G\,\alpha]$, sending a stream of generations of streams into a stream of generations of their data. To define it, we denote the k^{th} element of the argument to `join` by $gs_k = \{d_{k,0}, d_{k,1}, \ldots, d_{k,n}\}$ of type $G\,[G\,\alpha]$. Each $d_{k,i}$ is, in turn, a stream $[g^k_{i,0}, g^k_{i,1}, g^k_{i,2}, \ldots] : [G\,\alpha]$. We invert the structure of gs_k using a function $\texttt{transpose} : M[\alpha] \to [M\,\alpha]$. This function generalises matrix transposition on square arrays (type $[[\alpha]] \to [[\alpha]]$) to arbitrary monads M, and the generality allows us to abstract away from Spivey's bags and consider more exotic inner data-structures.[3]

$$\texttt{transpose}\,xs = \texttt{fmap}\,\texttt{head}\,xs :: \texttt{transpose}\,(\texttt{fmap}\,\texttt{tail}\,xs).$$

The transpose produces a stream of generations of generations (type $[G\,(G\,\alpha)]$). If we join each of the elements, we will have a stream $[D_{k,0}, D_{k,1}, D_{k,2}, \ldots] : [G\,\alpha]$ (see Figure 4.4), and thus, the shift function of Figure 4.3 will make sense. Each row is shifted relative to its predecessor by prepending the 0 generation, and the columns are combined by taking their sum.

The type of streams now constitutes a monad (see Spivey [92] for details). The fact that we have a monad affords us a tight integration with the host language in the following sense: we can lift arbitrary functions in the host language to functions on streams, and

[3] The operator `::` is *cons* for lists and streams.

$$
\begin{aligned}
&\texttt{map join}\,(\texttt{transpose}\,\{d_{k,0}, d_{k,1}, \ldots, d_{k,n}\}) \\
=&\texttt{map join}\left(\texttt{transpose}\left\{\begin{array}{cccc}
[g^k_{0,0}, & g^k_{0,1}, & g^k_{0,2}, & \ldots] \\
[g^k_{1,0}, & g^k_{1,1}, & g^k_{1,2}, & \ldots] \\
& \vdots & & \\
[g^k_{n,0}, & g^k_{n,1}, & g^k_{n,2}, & \ldots]
\end{array}\right\}\right) \\
=&\begin{bmatrix}
\texttt{join} & \{g^k_{0,0}, & g^k_{1,0}, & \ldots, & g^k_{n,1}\}, \\
\texttt{join} & \left\{g^k_{0,1}, \right. & g^k_{1,1}, & \ldots, & \left. g^k_{n,1}\right\}, \\
\texttt{join} & \{g^k_{0,2}, & g^k_{1,2}, & \ldots, & g^k_{n,2}\}, \\
& \vdots & & &
\end{bmatrix} \\
=&[D_{k,0}, D_{k,1}, D_{k,2}, \ldots]
\end{aligned}
$$

Figure 4.4: `transpose` and `join`

combine one stream $xs : [G\,\alpha]$ with another stream $xs' : \alpha \to [G\,\beta]$ which depends, via arbitrary computations, on each individual element of xs.

There is further algebraic structure in the form of a monoid:[4] streams can be summed by summing corresponding generations, an operation whose identity is the infinite stream of empty generations.

4.5 Case-analysis

Our algebra allows us to partition our domain into discoverer streams according to our own insight into a problem, and compose them in a way that reflects the typical reasoning patterns found in the domain. For incidence reasoning, we divide the domain into our five kinds of data discoverer, and combine the discoverers using our rules.

We wanted more than this, however, because when it comes to theorem-proving, the data is further partitioned as we perform case-analyses on disjunctions. In proof-search, when we encounter a disjunction, we will want to branch the search and associate data in each branch with its own disjunctive hypothesis.

Ideally, we want to leave such case-splitting as a book-keeping issue in our algebra,

[4] A structure with an associative operation and identity.

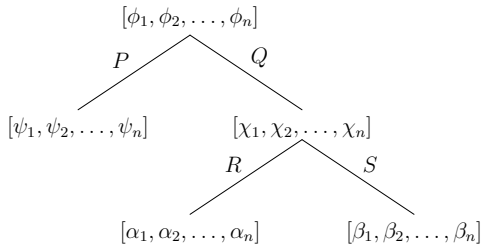


Figure 4.5: A simple tree branching on case-splits

and so integrate it into the composition algorithm. Streams must then record a context for all of their data, and this context must be respected as the streams are combined.

Our definitions give us the flexibility to implement this. We can choose a data-structure other than bags for the generations, and to solve the problem of case-analysis, we have chosen to implement the generations as *trees*.

4.5.1 Trees

Each tree represents a generation of data partitioned according to case-splits. Each node in a tree is a bag of data discovered in that generation under a disjunctive hypothesis. Branches correspond to case-splits, with each branch labelled for the disjunct on which case-splitting was performed. The branch labels along any root-path therefore provide a context of disjunctive hypotheses for that subtree. For example, the tree in Figure 4.5 might represent formula 4.2, made by case-analysing $P \vee (Q \wedge (R \vee S))$. Our goal is to discover data which hold when the case-splits are eliminated.

$$\begin{aligned} \phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \wedge (P \implies \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n) \\ \wedge \left(\begin{array}{r} Q \implies \chi_1 \wedge \chi_2 \wedge \cdots \chi_n \wedge (R \implies \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n) \\ \wedge (S \implies \beta_1 \wedge \beta_2 \wedge \cdots \wedge \beta_n) \end{array} \right). \end{aligned} \tag{4.2}$$

4.5.1.1 Combining Trees

The principal operation on trees is a sum function which is analogous to the append function for lists and streams, combining all data from two trees. The simplest way to combine two trees, one which yields a monad, has us nest one tree in the other. That is, we replace the leaf nodes of one tree with copies of the other tree, and then flatten. For definiteness, and to ensure associativity, we always nest the right tree in the left.

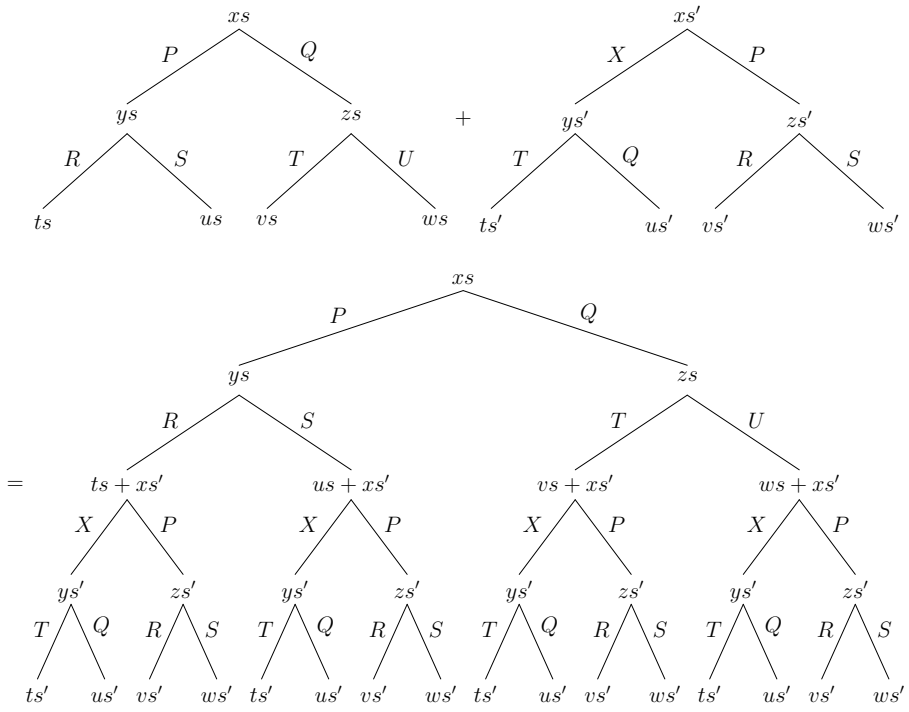


Figure 4.6: Combining trees by nesting

See Figure 4.6.

Even if we only had a constant number of case-splits, successive combining in this manner would yield trees of arbitrarily large topology, which is clearly not desirable. As such, we need some way to simplify the resulting trees. Our hard constraint is that we must not lose any information: if data is deleted from the tree, it must be because it is trivially implied by other data in the tree. Our softer constraint, which we do not attempt to formally define, is that the topologies should represent a reasonably efficient partitioning of the proof-space according to the combination of case-analyses.

Our first means of simplification is a form of weakening. If we have tree t which branches on a disjunctive hypothesis and which contains a subtree t' branching on that same hypothesis, then the root path is of the form

$$P \Longrightarrow \cdots \Longrightarrow Q \Longrightarrow \cdots \Longrightarrow Q \Longrightarrow \phi.$$

We eliminate the redundant antecedent by folding t' into its immediate parent. Any siblings of t' whose branch labels are not duplicated along the root path are then discarded. They will be left in the other branches of t.

The situation is shown in Figure 4.7, where we have coloured duplicate hypotheses along root paths. In the result, we fold the marked subtrees into their parents, and

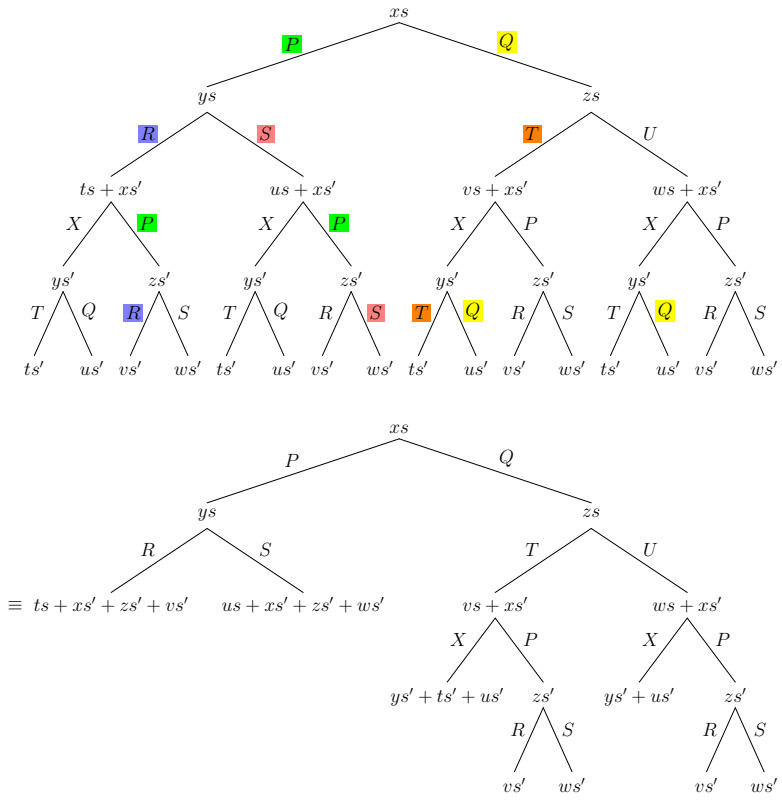


Figure 4.7: Weakening

discard the siblings. Notice that no data has been lost, and no individual bags at tree-nodes have been changed: the only operations we use here are topological changes to the tree, and bag union $(+)$.

Our next simplification allows us to delete a subtree t if its branch case is already considered by an uncle[5] t'. All theorems of t will appear in t' where the topology will have been simplified in our weakening step. The situation is shown in Figure 4.8. The P branch on the left-hand side is uncle to the P branches on the right. These latter branches are therefore pruned.

Finally, we can promote data that appears at all branches. In Figure 4.8, the xs' bag of theorems appears in every node at the same level, and so can be promoted into the parent, corresponding to disjunction elimination.

Our final tree is shown in Figure 4.9. We have described our simplification in several steps, though they can be combined into a single pass of the left hand tree. That said, with a single pass, we have not found a way promote *all* data. If we had, the replicated us' bag in the bottom right branches of the tree could be promoted into the Q branch.

[5]The sibling of a parent node.

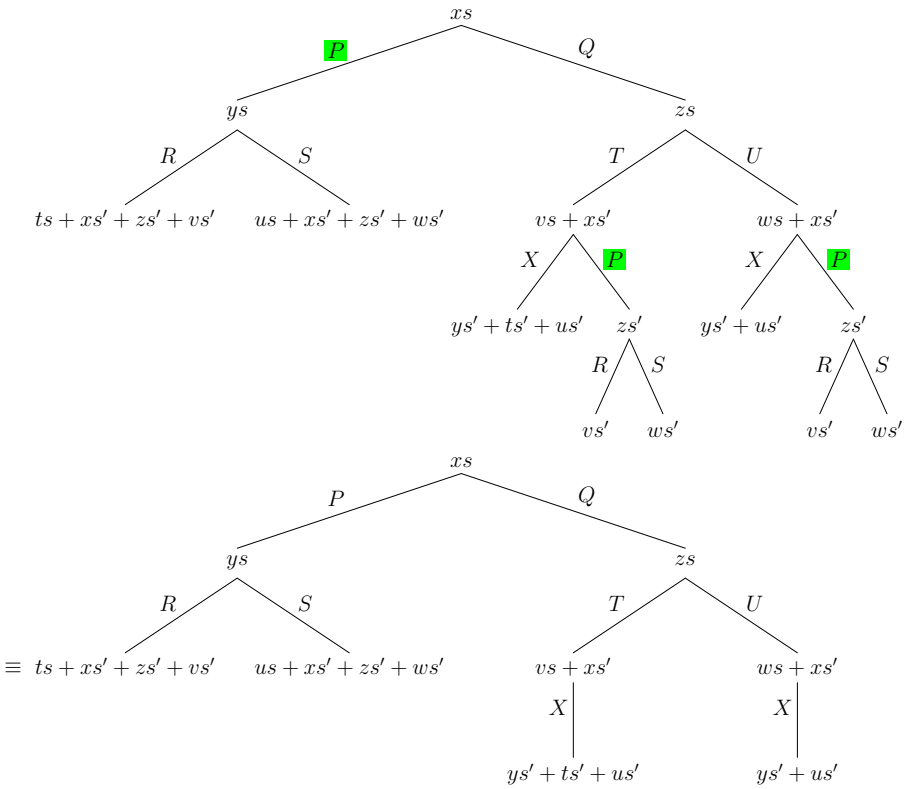


Figure 4.8: Removing redundant case-splits

We will not elaborate on this implementation issue.

4.5.1.2 Joining Trees

Our function which combines trees is a sum function, which has an identity in the empty tree containing a single empty root node. This is enough structure to define a `join` function analogous to list and stream concatenation. Suppose we are given a tree t whose nodes are themselves trees (so the type is `Tree` (`Tree` α)). Denote the inner trees by $t_0, t_1, t_2, \ldots, t_n$: `Tree` α. We now replace every node of t with an empty bag, giving a new tree t' : `Tree` α. We can now form the sum

$$t' + t_0 + t_1 + t_2 + \cdots + t_n.$$

The resulting tree will then contain discovered data which respect disjunctive hypotheses from their place in t and from their respective inner-trees.

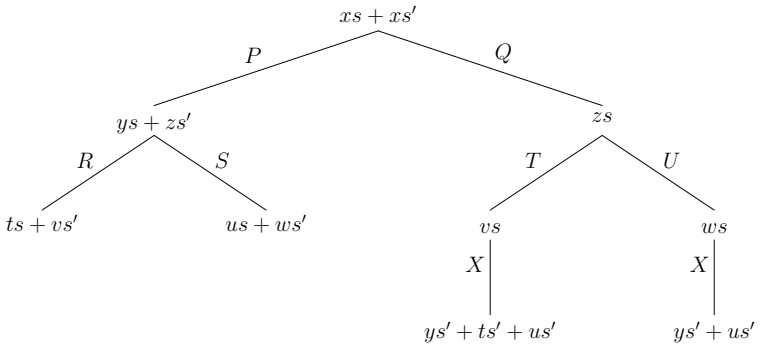


Figure 4.9: Promoting common data

4.6 Additional Primitives and Derived Discoverers

Using trees as generations, we have a stream data-type to represent a discoverer searching a space which has been partitioned according to case-splits. Since we do not specify what sort of data we are considering, our discoverers have polymorphic type. We alias the type as `discoverer` α.

As we described in §4.4.3, these discoverers form a monoid. Operationally, the sum function runs two discoverers in parallel, respecting their mutual case-splits, while our identity discovers nothing but empty generations.

We now explain additional functionality, some of which relates specifically to theorem proving and some of which more generally handles filtering. The following material, so far as we are aware, is new.

4.6.1 Case-splitting

Case-splits are introduced by our function `disjuncts`, which is a discoverer parameterised on arbitrary sequents. Here, `disjuncts`$(\Gamma \vdash P_1 \vee P_2 \vee \cdots \vee P_n)$ is a stream containing a single tree with n branches from the root node. The i^{th} branch is labelled with the term P_i and contains the single sequent $\Gamma \cup \{P_i\} \vdash P_i$. This process can be undone by flattening trees using `flatten`, which discharges all tree labels and adds them as antecedents in the right hand side of all sequents.

4.6.2 Delaying

A generation can be "put off" to a later generation using the function `delay`. In terms of streams, this function has the trivial definition

$$\texttt{delay}\ xs = \emptyset :: xs.$$

The use of this function is actually essential when writing mutually recursive discoverers. If one discoverer outputs data based on data output by another discoverer, and *vice versa*, then one of the two discoverers must be delayed to avoid deadlock.

4.6.3 Filtering

In many cases, we will not be interested in all the outputs generated by a discoverer. Fortunately, filtering is a library function for monads with a zero element, and can be defined in terms of the usual bind operation ($\triangleright$). We give definitions for both ($\triangleright$) and `filter` as follows:

$$xs \triangleright f = \texttt{join}\,(\texttt{fmap}\ f\ xs)$$

$$\texttt{filter}\ p\ xs = xs \triangleright (\lambda x.\ \texttt{if}\ \ p\ x\ \texttt{then return}\ x\ \texttt{else}\ 0).$$

More challenging is a function to perform something akin to *subsumption*. The idea here is that when a sequent is discovered which "trivially" entails a later sequent, the stronger sequent should take the place of the weaker. This is intended only to improve the performance of the system by discarding redundant search paths based on subsumed sequents.

We can generalise the idea to arbitrary data-types, and parameterise the filtering by any partial-ordering on the data, subject to suitable constraints. One intuitive constraint is that a stronger item of data should only replace a weaker item so long as we do not "lose" anything from later discovery. Formally, we require that any function f used as the first argument to `fmap` is monotonic with respect to the partial-order. That is, if $x \leq y$ then $f(x) \leq f(y)$.

We then implement "subsumption" as the transformation `maxima`. This transforms a discoverer into one which does two things: firstly, it transforms every individual generation into one containing only maxima of the partial-order. Secondly, it discards data in generations that is strictly weaker than some item of data from an earlier generation. To handle case-splits, we assert that data higher up a case-splitting tree is always considered stronger than data below it (since it carries fewer case-splitting assumptions).

4.6.3.1 More Sophisticated Filtering

There are two pieces of missing optimisation. Suppose we have data x and y where $x \leq y$ and a stream $xs = [\{x\}, \{y\}]$. That is, xs represents a discoverer which produces two generations of data. In the first generation is the single datum x and in the second is the single datum y. Now suppose we have a function f which happens to have the mappings

$$x \mapsto \left[\{\}, \{\}, \{\}, \{\}, \{x'\}\right]$$
$$y \mapsto \left[\{y'\}\right].$$

Here, y is sent immediately to the new datum y', while the image x' of x takes some time to generate, represented here by a succession of four empty generations. Thus,

$$xs \rhd f = \left[\{y'\}, \{\}, \{\}, \{\}, \{x'\}\right].$$

What happens when we want to take the maxima of this stream? To do this, f must be monotonic, and to verify this, we need some way to partially order the images of f, which means knowing more generally how to partially order discoverers.

In general, if two discoverers xs and ys are such that $xs \leq ys$, what should we say this implies about the ordering of the items discovered by each? We should be able to say at least this much: for every w in the first n generations of xs, there should exist some z in the first n generations of ys such that $w \leq z$. Thus, ys discovers data which is at least as strong as the data of xs, and does so at least as early in terms of the number of generations.

So if by monotonicity we require that $[\{\}, \{\}, \{\}, \{\}, \{x'\}] \leq [\{y'\}, \{\}]$, then we further require that $x' \leq y'$. Therefore:

$$\texttt{maxima}\ xs \rhd f = \texttt{maxima}\ \left(\left[\{y'\}, \{\}, \{\}, \{\}, \{x'\}\right]\right) = \left[\{y'\}, \{\}, \{\}, \{\}, \{\}\right].$$

And here we have a problem: the delayed and potentially slow computation of x' was wasted. Once y was discovered, further discovery based on x should have been halted. This does not happen in our implementation, and leads to a great deal of wasted effort. Consider the fact that every time we successfully apply Rule `colcol` to take the union of sets S and T, all further discovery based on S and T should be abandoned in favour of discovery using $S \cup T$. A similar issue applies to the discovery of equalities: as new equalities are discovered, they should rewrite all other data and ongoing discovery based on unwritten data should be abandoned.

A second piece of potentially desirable functionality is a form of *memoisation*. Suppose two generations of a discoverer are evaluated to $[\{x\},\{y\}]$ where $x \leq y$. With y evaluated, we would probably prefer any reevaluation of this discoverer to actually replace x, yielding $[\{y\},\{\}]$.

This additional functionality has not yet been implemented, and to get it to work, we might have to significantly modify the underlying data-structures for our streams. We leave such possibilities to future work.

4.6.4 Accumulating

We supply an accumulation function which is similar to the usual `fold` function on lists and streams. This threads a two-argument function through the data in a stream, starting with a base-value, and folding each generation down to a single intermediate datum. Thus, we have:

$$\begin{aligned}\texttt{accum}\,(+)\,0\,[\{1,2\},\{3,4\},\{5\},\{6,7,8\}] &= [\{0+3\},\{3+7\},\{10+5\},\{15+21\}]\\ &= [\{3\},\{10\},\{15\},\{36\}].\end{aligned}$$

One useful case of this allows us to gather up all data discovered so far in a single collection. If the collection is finite lists, we just use $\texttt{accum}\,(\lambda xs.\,\lambda x.\,x :: xs)\,[]$.

4.6.5 Deduction

Direct deduction, or *modus ponens*, is the basis of forward-chaining and we provide two main ways to lift it into the type of discoverers.

We first have to deal with *exceptions*. In HOL Light, the *modus ponens* inference rule will throw an exception if its arguments are of incorrect form, so we first redefine `fmap` to filter thrown exceptions out of the discovery. It is generally undesirable to let exceptions propagate upwards, since this would lead to an entire discoverer being halted.

With `fmap` redefined as `fmap'`, we can define functions `fmap2'` and `fmap3'` which lift two and three-argument functions up to the level of discoverers, also filtering for exceptions.

$$\texttt{fmap}'\,f\,xs = xs \rhd (\lambda x.\,\texttt{try return}\,(f\,x)\ \texttt{with}\ \_ \to 0)\,xs$$

$$\texttt{fmap2}'\,f\,xs\,ys = \texttt{fmap}\,f\,xs \rhd (\lambda f.\,\texttt{fmap}'\,f\,ys)$$

$$\texttt{fmap3}'\ f\ xs\ ys\ zs = \texttt{fmap}\ f\ xs \rhd (\lambda f.\ \texttt{fmap2}'\ f\ ys\ zs).$$

With these, we can define the following forward-deduction functions:

$$\begin{aligned}
\texttt{chain1}\ imp\ xs &= \texttt{fmap2}'\ \texttt{MATCH_MP}\ (\texttt{return}\ imp)\ xs \\
\texttt{chain2}\ imp\ xs\ ys &= \texttt{fmap2}'\ \texttt{MATCH_MP}\ (\texttt{chain1}\ imp\ xs)\ ys \\
\texttt{chain3}\ imp\ xs\ ys\ zs &= \texttt{fmap2}'\ \texttt{MATCH_MP}\ (\texttt{chain2}\ imp\ xs\ ys)\ zs \\
\texttt{chain}\ imps\ xs &= imps \rhd (\lambda imp.\ \texttt{if is_imp}\ imp \\
&\qquad \texttt{then chain}\ (\texttt{fmap}\ (\texttt{MATCH_MP}\ imp)\ thms)\ thms \\
&\qquad \texttt{else return}\ imp).
\end{aligned}$$

The function `is_imp` is a standard HOL Light function which returns `true` if the right hand side of its sequent argument is an implication, while `MATCH_MP` is the matching *modus ponens* derived inference rule

$$\frac{\Gamma \vdash P \implies Q \quad \Delta \vdash P'}{\Gamma \cup \Delta \vdash Q'}\ \texttt{MATCH\_MP}$$

where $P' = P[\theta]$ and $Q' = Q[\theta]$ for some substitutiton θ.

Thus, `chain1` takes a sequent $\Gamma \vdash P \implies Q$ and applies *modus ponens* across a discoverer of antecedent sequents. The function `chain2` takes a sequent of the form $\Gamma \vdash P \implies Q \implies R$ and applies *modus ponens* across two discoverers of antecedents. The function `chain3` takes a sequent of the form $\Gamma \vdash P \implies Q \implies R \implies S$ and applies *modus ponens* across three discoverers of antecedents. The final, more general `chain` function, recursively applies implication sequents with arbitrary numbers of curried antecedents from the discoverer *imps* across all possible combinations of antecedents from the discoverer *xs*.

Note that the discoverers `chain1`, `chain2` and `chain3` will not necessarily try all combinations of theorems from their arguments. They fail opportunistically, attempting *modus ponens* on each sequent from the first argument, and *only if it succeeds*, attempting *modus ponens* on sequents from the second argument. This is a happy feature of the data-driven semantics of monads (but see §4.9 for its drawbacks).

It is therefore sensible to order the antecedents of the implication according to how likely they are to succeed. Antecedents which rarely succeed should appear early to guarantee a quick failure. In our use case, for example, we apply Rule `coleq` from §4.2.1 with the collinear antecedents before the non-collinear antecedent, since there are generally many more non-collinear sequents to choose from than there are collinear sequents.

4.7 Integration

It is straightforward to integrate our discoverers with the rest of HOL Light's proof tools. We can, for example, lift term-rewriting to the level of discoverers simply by lifting HOL Light's rewriting functions with `fmap` and its derivatives.

To use our discoverers in declarative proofs, we introduce two Mizar Light primitives. The first, `obviously`, is used to assist any step in a declarative proof via a discoverer.[6]

$$\texttt{obviously} : (\texttt{discoverer thm} \to \texttt{discoverer thm}) \to \mathit{step} \to \mathit{step}.$$

The expression `obviously` f transforms a step into one which lifts the step's justifying theorems into a discoverer, applies the transformation f to this discoverer, and then retrieves all discovered theorems to a certain depth. These are then used to supplement the step's justification.

By allowing f to have type `discoverer thm` $\to$ `discoverer thm`, we can use a discovery pipeline via function composition to justify a declarative step. For example, later in our formal development we will introduce an incidence discoverer `by_incidence`. We also have a function `split` which recursively breaks conjunctions and splits disjunctions across all sequents found by a given discover. Finally, we have a rule `add_triangles` which introduces non-collinearity sequents from sequents about points inside and outside triangles. The three functions can be pipelined in a single step by writing

$$\texttt{obviously}\,(\texttt{by\_incidence} \circ \texttt{add\_triangles} \circ \texttt{split})\ldots.$$

Next, we introduce a primitive `clearly`. This has the same type as `obviously`, but rather than collecting all discovered sequents, it searches specifically for the formula to be obtained by a Mizar Light step (in the case of a `qed` step, this is the goal formula). When the `clearly` primitive is used, it leaves the basic Mizar Light justification tactic with a trivial proof.

Finally, we have introduced a theorem-tactic `discover_tac` with type

$$\texttt{discover\_tac} : (\texttt{discoverer thm} \to \texttt{discoverer thm}) \to ([\texttt{thm}] \to \texttt{tactic}) \to \texttt{tactic}.$$

[6] The type `thm` is the type of HOL Light sequents. The type `step` is the type of Mizar Light steps.

As with `obviously` and `clearly`, we take a transformation of discoverers. Then when we apply `discover_tac` f *tac*, all goal hypotheses are fed to f. The resulting discovered sequents are then supplied to the function *tac* (often just `MESON`).

4.7.1 Concurrency

In §4.3.1, we hinted at the possibility of making our algorithm concurrent and collaborative. The implementation is straightforward with streams.

So that discoverers can incorporate the user's manual proof efforts, they need a way to inspect the proof context. We use a simple new primitive discoverer `monitor` to allow this. As we stated in our introduction in §4.4.2, primitive discoverers can generate their outputs in whatever manner they want. Our `monitor` just regularly examines the proof context and outputs its hypotheses. We ensure that only unique hypotheses are ever generated by using the function `maxima`.

For the user's inspection, discovered theorems are output to the terminal one-by-one, simply by iterating over the streams with an ordinary print function in a separate thread. Additionally, this thread updates a reference cell `the_facts` which holds all theorems discovered in that thread so far. The cell can be deferenced in the interactive proof, so that the user can make use of the results of concurrent discoverers as they come in.

To complete the architecture, we provide signalling commands that can tell a discovery thread to pause, thereby freeing up CPU resources, to resume from a pause, or to change to an alternative theorem discoverer, so that the thread does not have to be killed.

We should admit to a caveat here. Generations, as we have implemented them, are not generally usable across the branches of case-splits and subproofs of a declarative verification. When the user completes a branch or subproof, some of their assumptions will no longer be in force. Thus, the user can no longer apply data generated from those assumptions to justify the next proof step. If they try to do so, Mizar Light will throw an error when the step command is processed, as part of the basic validity checking of the tactic system. Instead, the user must manually reset the discoverers after each case-split and subproof, to throw out the assumptions no longer in force. This is far from ideal, but as yet, our implementation does not address the issue.

4.7.2 Dependency Tracking

While writing a proof, we would normally start our incidence discoverer concurrently, which would generate sequents as we worked. The discoverer would consider all of our goal hypotheses, and would typically find a large number of sequents, most of which were not needed in the proof. Of those which were needed, we wanted to know the specific subset of hypotheses from which they were derived. This is desirable when it comes to proof-replay, when we would like the discoverer to work more efficiently with just the hypotheses it needs. But it is also desirable in terms of writing declarative proofs: we want to write the dependent hypotheses directly into the proof script, so they are apparent to the reader. To achieve this, we need our discoverers to track hypotheses.

A nice feature of monads in functional programming is that, when defined in the form of a *transformer*, the extra book-keeping can be added in a clean and modular way. In this case, we create a generic *writer transformer*.

Writer is a monad with extra functions for *writing* values from any monoid during a computation, and for running computations to retrieve the final written values. It provides a modular way to introduce things such as *logging* into computations. So if we have a computation which writes the value `"Hello"` and returns the value 1, and another computation which writes the value `" world!"` and returns the value 2, then when we lift addition over the two computations, we have a computation which returns the value 3 and writes `"Hello world!"`. The monoid here is strings with the append operation.

Writer can be defined as a transformer which, when applied, is made to write values inside any other monad. We made our writer work on a monoid of hypothesis sets with the union operation. These sets contain the dependent hypotheses from the proof context for every discovered sequent, and they automatically accumulate as discoverers are combined. We do not need to write any special logic for this. All the details are handled generically by the writer transformer.

All we need to do is extend the `monitor` discoverer to initialise the dependent hypotheses sets. So now, every time `monitor` pulls a hypothesis from the proof context, it must also write the hypothesis as a dependency on further computation. The dependencies then automatically propagate.

4.8 Implementation Details

The following section describes some technical challenges we faced. We include it for the sake of honesty, since the presentation of the ideas so far ignores all the bumps that thwart a nice clean implementation in Ocaml. We assume some knowledge of the Ocaml language here.

We have implemented a general monad library in Ocaml, providing a signature for the minimal implementation of an arbitrary monad, and a *functor*[7] to turn this minimal implementation into a richer monad interface with derived functions. Monad transformers are functors between minimal implementations of monads. The stream monad itself is a transformer from an arbitrary monad of generations. If we want to use the bag implementation, we can just supply a module of bags to this transformer. If we want to use our case-splitting implementation, we supply our module of trees.

These transformations can be stacked. In fact, in our final implementation, we first apply our writer transformer to a monoid of hypotheses sets. The writer transformer is then applied to our tree monad. Finally, the tree monad is applied to our stream monad to produce a stream which discovers theorems, handles case-splits and tracks dependent hypotheses.

4.8.1 Implementation Issues

As of writing, HOL Light relies heavily on modifications to Ocaml's preprocessor, one of which forces Ocaml's lexer to treat any lexeme which contains more than one upper-case letter as being a lower-case identifier. This plays havoc with the CamelCase naming convention now being adopted in Ocaml's "Batteries Included" library, since Ocaml's parser expects all module names to be uppercase. To circumvent this, we supply our own preprocessor which allows for lower-case identifiers in module names. This is only intended as a hack over a hack until we find a robust solution.

Concurrency often raises thorny issues in working code, and some HOL Light functions were not developed with concurrency in mind. The basic `MESON` tactic, which we rely on throughout our verifications, is not thread-safe, and it must be avoided when defining discoverers.

Another issue with threading concerns UNIX signals. The standard way to interact

[7] All uses of the term *functor* here refer to a feature of the Ocaml module system. Though a term borrowed from category theory, it is not in the same context as *monad* as we have used it in this chapter.

with HOL Light is to run intractable decision procedures on problems. If these procedures fail, the user interrupts them at the Ocaml top-level by sending `SIGINT`. If the user is running a concurrent discoverer when they send this signal, it might interrupt the wrong thread, possibly leaving the discoverer with dangling resources. As a quick hack around this behaviour, we trap SIGINT in the discovery thread, pause, and emit it again.

Finally, Ocaml's lazy list library has some unexpected implementation choices. For instance, the list concatenation function *concat* : $[[\alpha]] \to [\alpha]$ is *strict* in its outer list, while the `take` function which takes a certain sized prefix of a given lazy list will always force the values in the prefix. These behaviours are generally inappropriate for lazy data-structures, and their use often leads to infinite looping when we come to write recursive functions which generate lazy lists. We have had to rewrite many of them.

Besides these issues, the syntax for lazy lists in Ocaml is cumbersome. Primitive list functions have to explicitly force lazy lists, while recursive lazy lists must be wrapped with a `lazy` keyword. One benefit, however, of Ocaml's lazy list implementation is that it detects when a recursive definition requires forcing a vicious circularity. Such a scenario arises when the computation of the first element of *xs* requires computing the first element of *ys*, and *vice versa*. Such situtations occur easily when writing discoverers in our algebra, and are fixed with choice uses of the `delay` function.

4.9 Applicative Functors

An *idiom F* or *applicative functor* [62] is a generalisation of a monad, which could be thought of as providing at least the ability to lift values with a function *pure* : $\alpha \to F\ \alpha$ and to lift a two argument function as we do with

$$\texttt{fmap2} : (\alpha \to \beta \to \gamma) \to F\ \alpha \to F\ \beta \to F\ \gamma.$$

This is actually quite limiting compared to the monad (see [84] for a detailed analysis). We can no longer write data-driven computations, such as our `chain` functions which automatically fail at the first non-matching antecedent.

That said, because it does not allow data-driven computation, an implementation of the applicative functor interface can potentially be more efficient than the implementation it derives as a generalisation of a monad. This was something we spotted when fixing

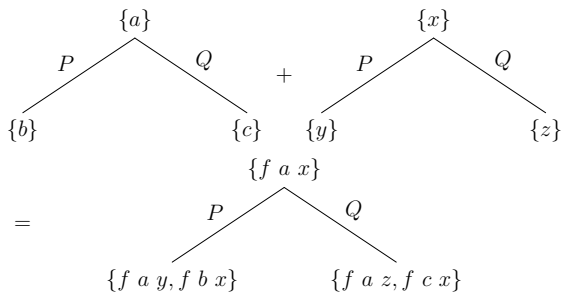


Figure 4.10: Applicative simplification

a performance issue with our trees.

In Figure 4.10, we show what happens when we lift a two argument function f over two trees with the same topology. The result is correctly simplified, so that it appears that the function f is evaluated exactly five times. But, bizarrely, when we came to profile our code, we found that the function f was evaluated *nine* times, once for each possible combination of data in the two trees. The data were discarded in simplification, but only after they were evaluated. This completely breaks the intended purpose of the trees, which is to partition the data so that the discarded computations never take place.

The inefficiency is actually unavoidable with the monad implementation. A computation in a monad produces structure dependent on data within another structure. When we lift a two argument function over two structures, it happens that the structure of the final value is independent of this data, but the independence cannot be guaranteed by the type system.

This is the advantage of the applicative functor. In defining it, we are not allowed to peek at the data inside our structures, so when lifting a two argument function, the type system can guarantee that the final structure can only depend on the structures of the two inputs.

For our trees in Figure 4.10, we know, without looking at the data, what the topology of the final result should be. We know, in advance, what simplifications should come in, and we therefore know, in advance, how many times the function f will be applied. But this knowledge is only guaranteed for applicative functors. When we use the derived monad implementation, f must be applied across all computations and only *then*, when all dependencies on the data have been take into account, can simplification take place.

A simple fix is to provide an explicit applicative implementation which will override the derived monad implementation. The resulting overridden interface is identical from the perspective of client code, but now, when the client does not require data driven

computations, such as with the applicative functions `fmap2`, `fmap3` and so on, the more efficient implementation is used.

4.10 The Problem Revisited

We now return to our original data-flow problem. In Figure 4.11,[8] we use our discovery algebra to capture the complex data-flow network from Figure 4.1. As we can see, the five kinds of data now correspond to five primitive discoverers. Rules are applied across the data by lifting those rules into the type of discoverers, and mutual dependencies of the network are captured by mutual recursion.

One advantage of our algebra is that it is almost trivial to refine the discovery system. For instance, we noticed that the network in Figure 4.1 has some redundancy: point-inequalities delivered from non-collinear sets by the rule `ncolneq` should not be used to try to infer *new* non-collinear sets. We eliminated this redundancy by splitting `neqs` into two discoverers, `neqs` and `neqs'`.

$$
\begin{aligned}
\mathit{non\_collinear} &= \texttt{maxima}\,(\texttt{filter}\ \texttt{is\_non\_collinear}\ \mathit{thms} \\
&\qquad + \texttt{fmap3'}\ \texttt{colncolncol}\ \mathit{collinear}\ (\texttt{delay}\ \mathit{non\_collinear})\ \mathit{neqs}') \\
\texttt{and}\ \mathit{neqs} &= \texttt{maxima}\,(\texttt{neqs'} \\
&\qquad + \texttt{fmap'}\ (\texttt{sum}\,(\texttt{conjuncts} \\
&\qquad\qquad (\texttt{chain1}\ \texttt{ncolneq}\ (\texttt{delay}\ \mathit{non\_collinear}))))) \\
\texttt{and}\ \mathit{neqs}' &= \texttt{maxima}\,(\texttt{filter}\ \texttt{is\_neq}\ \mathit{thms} \\
&\qquad + \texttt{sum}\,(\texttt{fmap2'}\ \texttt{colncolneq}\ \mathit{collinear}\ (\texttt{delay}\ \mathit{non\_collinear})) \\
&\qquad + \texttt{sum}\,(\texttt{conjuncts} \\
&\qquad\qquad (\texttt{chain1}\ \texttt{ncolneq}\ (\texttt{filter}\ \texttt{is\_neq}\ \mathit{thms})))).
\end{aligned}
$$

4.11 Conclusion and Further Work

We hope the reader agrees that, with our combinator language, we can write neatly and declaratively specified data-flows to search for theorems, where discovered theorems are fed back into the network. The basic combinators constitute a monad to give them a familiar and robust semantics, and our implementation in streams makes it easy for

[8] The inference rule `CONJUNCTS` sends a conjunctive sequent to a list of its conjuncts.

```
    sum = foldr (+) 0 ∘ map return
    conjuncts = fmap' CONJUNCTS

by_incidence thms =
  let rec collinear =maxima (filter is_collinear thms
                        + fmap3' colcol (delay collinear) (delay collinear)
                                neqs)
   and non_collinear =maxima (filter is_non_collinear thms
                        + fmap3' colncolncol collinear (delay non_collinear)
                                neqs)
       and eqs =filter is_eq thms
                        + maxima(sum (fmap3' coleq
                            collinears collinear non_collinear))
         and neqs =maxima(filter is_neq thms
                        + sum (fmap2' colncolneq collinear
                                (delay non_collinear))
                        + sum (conjuncts (chain1 ncolneq non_collinear)))
       and planes =maxima (filter is_plane thms
                        + fmap3' planeplane (delay planes) (delay planes)
                                non_collinear
                        + fmap3' colplaneplane collinear (delay planes) neqs
                        + fmap2' colcolplane collinear collinear
                        + fmap' colplane collinear
                        + fmap' ncolplane non_collinear)
    in collinear+non_collinear + eqs + neqs + planes
```

Figure 4.11: Incidence discovery

searches to be run concurrently following the steps of a declarative proof, with the results made available to verify subsequent steps. This feedback makes the automation potentially collaborative, with the user and proof assistant sharing data as they progress towards the goal.

The theoretical underpinnings of our tree data-structure are left to further investigation. In particular, we would like to know more about the properties of our simplification algorithm, and establish, for instance, that a tree's theorems are trivially entailed by the theorems of its simplified tree. In other words, we would like to establish that simplification does not lose interesting theorems.

We regard this as beyond the scope of our verification efforts, though we point out that the incidence discoverer will be battle tested in later chapters, and that there have been no surprises which would indicate a mistake in our formulation.

We still need to devise a way to integrate proper subsumption into our algebra as mentioned in §4.6.3. A particular challenge here is finding an effective way to integrate normalisation with respect to derived equalities. We suspect any solutions here will require modifying our basic tree data-structures.

Our discovery language does not yet provide functions for more powerful first-order and higher-order reasoning. Our domain was a relatively simple combinatorial space of concrete incidence claims, but in the future, we would like to be able to apply the system to inductive problems, having it speculate inductive hypotheses and infer universals. Since the basic discovery data-type is polymorphic and not specific to theorem-proving, we hope that lemma speculation will just be a matter of defining appropriate search strategies. We would also like to handle existential reasoning automatically, and we are still working on a clean way to accomplish this.

Finally, we would like to consider abstracting the search algebras in the direction that Spivey has recently taken [92]. This allows us to abstract over various search strategies, including iterative deepening, and potentially provides a way to cleanly deal with a strategy based on subsumption. The advantage of following Spivey here is that his approach has very well developed theoretical underpinnings, which we believe are crucial when faced with the sorts of complexities found in algebras for unbounded search.

We leave the detailed evaluation of our incidence reasoner to the next chapter, where we shall apply it to three proofs from Hilbert's text.

Chapter 5

Elementary Consequences in Group II

We now come to verify some theorems of Group II, the only theorems in the first two groups which have prose proofs in the tenth edition of the *Grundlagen der Geometrie*. Each proof uses Axiom II, 4, which, as we explained in Chapter 3, carries several incidence preconditions. Establishing these preconditions made up the bulk of the effort in our manual verifications [86] and in the verifications of Meikle and Fleuriot [66].

In the last chapter, we described some automation to handle the incidence reasoning. An aim of the present chapter is to see how much this automation can help us shorten our verifications, make it easier to develop them and make it possible to find alternative proofs. Our ideal is for the verification steps to match the steps of an ideal prose proof, modelled here by Hilbert's own arguments, which focus on defining and exhibiting the diagram needed to establish a result, rather than discharging tedious incidence preconditions.

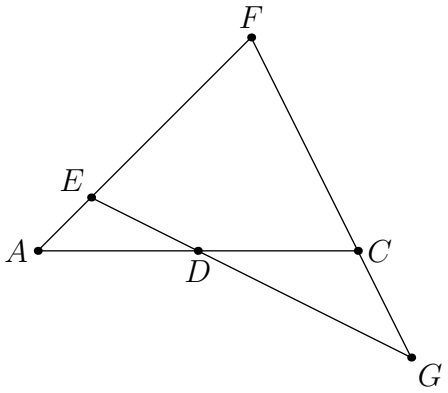
When writing our verifications, we tried our best to follow the specific steps in Hilbert's prose, and in this chapter, we shall directly compare our verifications with the originals. As we shall see, this allows us to comment on the prose in great detail.

5.1 THEOREM 3

Hilbert's first result tells us that there is a point between any two others.

> THEOREM 3. For two points A and C there always exists at least one point D on the line AC that lies between A and C.
>
> PROOF. By Axiom I, 3 there exists a point E outside the line AC and by Axiom II, 2 there exists on AE a point F such that E is a point of the segment AF. By the same axiom and by Axiom II, 3 there exists on FC a

point G that does not lie on the segment FC. By Axiom II, 4 the line EG must then intersect the segment AC at a point D.



[42, p. 6]

The only incidence axiom Hilbert appeals to in this proof is Axiom I, 3, and the only order axioms are Axiom II, 2 and Axiom II, 4. The proof depends on more axioms than this, but Hilbert's omission is consistent with a claim we made in Chapter 3: Hilbert generally only cites axioms which *introduce* points, omitting the others because he wants to focus on the steps which build up the diagrams. Here, we have Axiom I, 3 which can be used (indirectly) to obtain points off arbitrary lines, in this case the point E. We have Axiom II, 2 which is our "line-extension axiom", used here to first obtain the point F, and then to obtain the point G. Finally, we have Axiom II, 4 which finds "exit points" of a line passing through a triangle, in this case, the point D.

THEOREM 3 was not proven in the first edition of the *Grundlagen der Geometrie*; it was an axiom. That it turns out to be redundant might still come as a surprise, when we realise that we are obtaining such a simple linear result from a one-dimensional order axiom (II, 2) and a two-dimensional order axiom (II, 4). This is the situation in all three proofs we consider in this chapter. We will be proving one-dimensional results by obtaining two-dimensional figures and applying Pasch's axiom.

5.1.1 Verification

Our HOL Light verification shown in Figure 5.1 improves hugely on our manual verification [86], which ran to twenty-two steps. Here, we have just five steps, and are *very* close to Hilbert's prose. We have just one extra step: our final `qed` eliminates the unwanted disjunct from Pasch's axiom.

```
assume A ≠ C
so consider E such that ¬(∃a. on_line A a ∧ on_line C a ∧ on_line E a)
    by (I, 2), (I, 3.2)                                                    0
obviously by_neqs consider F such that between A E F from 0 by (II, 2)     1
obviously by_neqs so consider G such that between F C G
    from 0 by (II, 2)                                                      2
obviously by_incidence so consider D such that
    (∃a. on_line E a ∧ on_line G a ∧ on_line D a)
    ∧ (between A D C ∨ between F D C)
    using K (MATCH_MP_TAC (A.1)) from 0, 1
obviously (by_eqs ∘ split) qed from 0, 1, 2 by (II, 1), (II, 3)
```

Figure 5.1: Verification of THEOREM 3

Note that the verification presented here is in its final form, packaged from an interactive verification that was developed and assisted by concurrent discoverers. We will recount the general way we used the concurrent discoverers to produce these final packaged versions in §5.3.2, where we consider the verification of THEOREM 5.

In the verification, we cite two axioms that Hilbert did not. His Axiom I, 3 can only be used *indirectly* to find a point off an arbitrary line. Strictly speaking, one must also appeal to Axiom I, 2 as we have done. We have also cited Axiom II, 1. This is only needed for the trivial matter of switching the order of the outer arguments to the `between` relation.

We reference three separate discoverers in our proof: `by_incidence`, `by_neqs`, and `by_eqs`. The first discoverer collects all five kinds of incidence sequent considered in the last chapter into a single discoverer, while the latter two just discover inequality and equality sequents respectively. The semantics of laziness makes this a genuine optimisation: if we only pull sequents from the `by_neqs` discoverer, no sequents will be pulled from the `by_eqs` or `by_planes` discoverers, as we can see by looking again at the dependencies in our data-flow diagram (Figure 4.1).

Finally, we have used `MATCH_MP_TAC`[1] to apply our version of Pasch's Axiom (A.1) formulated entirely in terms of points (see Appendix A). This is slightly ugly, but it helps `MESON` by directing it to the antecedents of the matched theorem. It has no other side effects that carry over to the remaining steps, and its use does not break the declarative style since we state the implicational theorem that we have matched against. Even so, such matching is an irritation, since the free variables in the matched theorem must be lined up with those in the goal, and it is easy to get the order mixed up. We shall deal with this matter in §5.2.1.

5.1.2 The Outer and Inner Pasch Axioms

THEOREM 3 was an axiom of the first edition of the *Grundlagen der Geometrie*, and the full investigation of such redundant axioms can be credited to Veblen and his supervisor E.H. Moore, who investigated ordered geometry based on axioms very similar to Hilbert's. However, if we look at Veblen's system, we see he chose a different rendering of Pasch's Axiom.

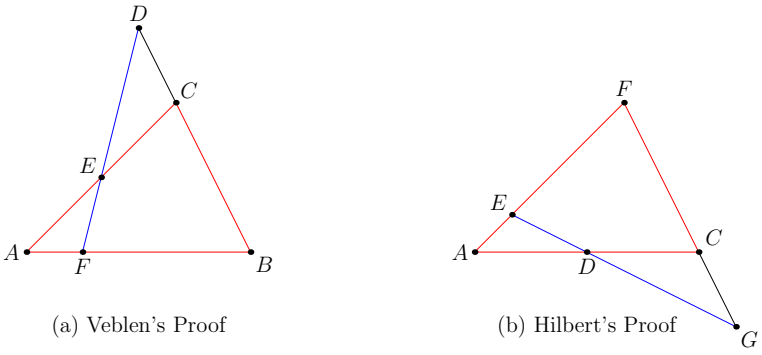
> "AXIOM VIII *(Triangle traversal axiom). If three distinct points A, B, C do not lie on the same line, and D and E are two points in the order BCD and CEA, then a point F exists in the order AFB, and such that D, E, F lie on the same line.*"
>
> [100, p. 355]

This form of the axiom, known as the *Outer Pasch Axiom*, is due to Peano. It is strong enough to replace Hilbert's own version, and has advantages in terms of incidence reasoning: it carries fewer incidence preconditions at the expense of one extra order condition, and the conclusion is not disjunctive, so we are saved the effort of eliminating an offending case.

A related axiom (which turns out to be weaker [78]), is also due to Peano. This is the *Inner Pasch Axiom*, which exchanges the roles of E and F in the Outer Pasch Axiom. It can be verified as:

$$\begin{aligned}\vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)& \\ \wedge\, \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ B& \\ \Longrightarrow\ \exists F.\ \exists a.\ \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F \wedge \texttt{between}\ A\ F\ C.&\end{aligned} \tag{5.1}$$

[1] This standard HOL Light function matches the conclusion of an implicational theorem with the goal formula, and then generates a new subgoal to prove the antecedent.



(a) Veblen's Proof (b) Hilbert's Proof

Figure 5.3: Veblen's versus Hilbert's Proof

It can now be seen that the diagram obtained in Hilbert's proof of THEOREM 3 is just obtaining the assumptions of this Inner Pasch Axiom. So when we verify Inner Pasch and use it as an alternative to Axiom II, 4, we get the verification shown in Figure 5.2, which does not need the final step we had before. In fact, had we spotted the factoring in our manual verification, we predict that we would have only needed eight, not twenty-two steps.

```
assume A ≠ C
so consider E such that ¬(∃a. on_line A a ∧ on_line C a ∧ on_line E a)
    by (I, 2), (I, 3.2)                                                    0
obviously by_neqs consider F such that between A E F from 0 by (II, 2)     1
obviously by_neqs so consider G
    such that between F C G from 0 by (II, 2)                              2
obviously by_ncols qed from 0, 1 by (II, 1), (5.1)
```

Figure 5.2: Verification of THEOREM 3 using the derived Inner Pasch Axiom

Veblen's diagram replaces the Inner Pasch Axiom with the Outer Pasch Axiom, but is otherwise very similar. If we take a relabelling $B \mapsto C, C \mapsto F, D \mapsto G$ and $F \mapsto D$, then we see that Veblen's second use of Axiom II, 2 (the line extension axiom) differs from Hilbert's by finding the point G on the other side of the segment CF (see Figure 5.3). Where Hilbert sets himself up to use the Inner Pasch Axiom, Veblen sets himself up to use the Outer Pasch Axiom:

> [...]. Between every two distinct points there is a third point.
>
> Proof. Let A and B be the given points [figure]. [...] there is a point E not lying on the line AB. By [the line extension axiom] points C and D exist, satisfying the order-relations AEC and BCD. Hence, by [the Outer Pasch Axiom], F exists in the order AFB.
>
> [100, p. 355]

In conclusion, for those of us who judge Hilbert's argument for THEOREM 3 to be gappy because of missing incidence reasoning, we offer a clean way to bring it up to more pedantic standards. Before THEOREM 3, we suggest one first derive the Inner Pasch Axiom (5.1), after which the proof follows more fluidly. This observation will be worth bearing in mind when we come to THEOREM 5 in §5.3, where we shall derive stronger versions of both the Inner and Outer Pasch axioms.

5.2 THEOREM 4

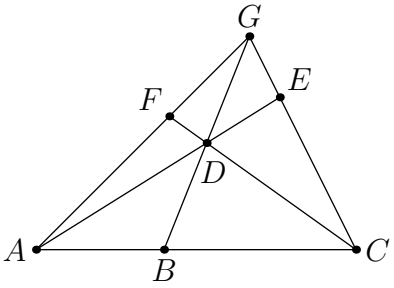
In this section, we review how we used our discovery tool in an exploratory fashion, as we examine Hilbert's THEOREM 4. This result was another axiom in the first edition of the *Grundlagen der Geometrie*, or more accurately, was incorporated into Axiom II, 3:

> "II, 3. *Of any three points situated on a straight line, there is always one and only one which lies between the other two.*"
>
> [41, p. 4]

The "only one" part is all that is retained in the tenth edition. The existence part is given in a proof which Hilbert credits to Wald.

> THEOREM 4. Of any three points A, B, C on a line there always is one that lies between the other two.
>
> PROOF. Let A not lie between B and C and let also C not lie between A and B. Join a point D that does not lie on the line AC with B and choose by Axiom II, 2 a point G on the connecting line such that D lies between B and G. By an application of Axiom II, 4 to the triangle BCG and to the line AD it follows that the lines AD and CG intersect at a point E that lies between C and G . In the same way, it follows that the lines CD and AG meet at a point F that lies between A and G.
>
> If Axiom II, 4 is applied now to the triangle AEG and to the line CF it becomes evident that D lies between A and E, and by an application of the

same axiom to the triangle AEC and to the line BG one realises that B lies between A and C.



[42, p. 7]

5.2.1 Discovering Applications of Pasch

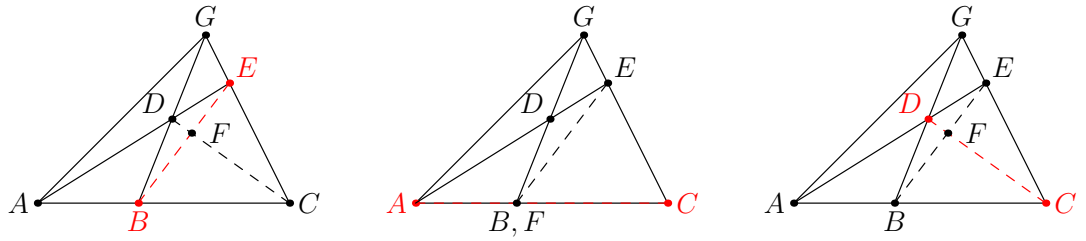
With Axiom II, 4 used a total of four times, and with the symmetry that appears in the diagram, we wanted to explore the proof of THEOREM 4 using our automated discoverers. Currently, our discoverers only tell us about the various incidence relations implied by our assumptions. This helps us discharge preconditions on Axiom II, 4, but does not tell us directly which instantiations of this axiom can be applied.

We cannot define a new discoverer which finds possible applications of Axiom II, 4 by using our simple forward-chaining primitives `chain1`, `chain2` and `chain3`. The problem is that our version of this axiom in point sets (3.11) has five free variables but its antecedents involving triangles and betweenness only fix three variables at a time. The remaining two variables can only be fixed by matches up to associativity and commutativity, which `MATCH_MP` does not support.

Instead, we defined a new discoverer `by_pasch` with a combination of ML and monad library functions. We filter for betweenness sequents from an existing discoverer and use these to discharge one of the preconditions of Axiom II, 4. We then reuse the discoverer `by_ncols` to discharge the triangle preconditions, manually handling the ordering of the free variables each time a precondition is eliminated.

The `by_pasch` discoverer therefore outputs sequents which are the conclusions of Axiom II, 4, telling us where the axiom can be applied. At the point in Hilbert's proof where the axiom is first used, the discoverer finds only one other possibility. Hilbert uses one after the other.

$$\exists F.\ (\exists a.\ \texttt{on line}\ C\ a \wedge \texttt{on line}\ D\ a \wedge \texttt{on line}\ F\ a)$$
$$\wedge\ (\texttt{between}\ A\ F\ B \vee \texttt{between}\ A\ F\ G).$$



(a) F is between B and E (b) F is between A and C (c) F is between C and D

Figure 5.4: Three possible applications of Axiom II, 4

$$\exists F.\ (\exists a.\ \texttt{on line}\ A\ a \wedge \texttt{on line}\ D\ a \wedge \texttt{on line}\ F\ a)$$
$$\wedge\, (\texttt{between}\ B\ F\ C \vee \texttt{between}\ C\ F\ G).$$

After Hilbert applies the first of these, another three possibilities arise, depicted in Figure 5.4.

(a) $\exists F.\ (\exists a.\ \texttt{on line}\ C\ a \wedge \texttt{on line}\ D\ a \wedge \texttt{on line}\ F\ a)$
$\wedge\, (\texttt{between}\ B\ F\ E \vee \texttt{between}\ E\ F\ G).$

(b) $\exists F.\ (\exists a.\ \texttt{on line}\ B\ a \wedge \texttt{on line}\ E\ a \wedge \texttt{on line}\ F\ a)$
$\wedge\, (\texttt{between}\ A\ F\ C \vee \texttt{between}\ A\ F\ G).$

(c) $\exists F.\ (\exists a.\ \texttt{on line}\ B\ a \wedge \texttt{on line}\ E\ a \wedge \texttt{on line}\ F\ a)$
$\wedge\, (\texttt{between}\ C\ F\ D \vee \texttt{between}\ D\ F\ G).$

Cases (a) and (c) of Figure 5.4 obtain the exact same point F, but in case (a) we should conclude that F is between B and E while in (c) we should conclude that F is between C and D. If we apply both cases of the axiom, we will know that F is between B and E and simultaneously between C and D. Either of these cases yields an alternative proof of the theorem which we describe in §5.2.3.

It would be surprising if case (b) got us *anywhere*. The only valid disjunct in its conclusion should put the point F between A and C, from which we could immediately conclude that $B = F$ and thus complete the proof. But this is all too easy. The truth of the matter is that incidence reasoning alone, according to our discoverers, cannot reject the impossible disjunct in the axiom's conclusion.

5.2.2 Verifying Hilbert's Proof

When we verified THEOREM 4 manually, it ran to sixty-nine steps. In this comparatively long verification, the structure of the basic prose argument is buried by incidence

arguments which do not illuminate anything. The arguments consist mostly of applications of our point set rules given in §3.5 with manual variable instantiations. We tried to use comments to show how the prose translated into the verification steps, but in the end, any claims of a faithful verification were weak.

But with our incidence automation, we have a verification in just thirteen lines, each readily understandable, and matching the prose very closely. The verification is shown in Figure 5.5, and we are again reasonably close to Hilbert's prose. The only warts are due to the `by_pasch` discoverer not handling the case-split in the conclusion of Axiom II, 4. This requires the elimination of a disjunct and the identification of the obtained point with an existing point. Both tasks are now handled in a subproof using a second discoverer, `by_eqs`.

We now review the verification. At the very start, we set our goal formula to be

$$(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ \Longrightarrow \texttt{between}\ A\ B\ C \vee \texttt{between}\ B\ A\ C \vee \texttt{between}\ A\ C\ B.$$

We then get to `assume`, just as Hilbert does, that C is neither between A or B nor A between B or C. This is a very natural way to express one's assumptions mathematically. It goes through because of the way Mizar Light implements the `assume` primitive: it first tries to prove the goal under the negation of our assumption. If this is successful, the assumption is used to rewrite the goal before being placed into the goal hypotheses. The upshot is that our goal formula gets rewritten to

$$(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \Longrightarrow \texttt{between}\ A\ B\ C.$$

Next, we look at the first two applications of Axiom II, 4. In our verification, we have not been able to apply this axiom in either the inner or outer forms, since we have only one order hypothesis. This means we are faced with having to eliminate a disjunct in the conclusion of Axiom II, 4. To do this, we find point equalities via the composed discoverer `by_eqs ∘ split`, which tells us that, in the offending cases, the point A lies between B and C or C lies between A and B. We have explicitly hypothesised against these two possibilities, and thus, the disjuncts can be eliminated.

```
assume ∃a. on_line A a ∧ on_line B a ∧ on_line C a                              0
assume A ≠ B ∧ A ≠ C ∧ B ≠ C                                                    1,2,3
assume ¬between A C B ∧ ¬between B A C                                          4
consider D such that ¬(∃a. on_line A a ∧ on_line B a ∧ on_line D a)
    from 1 by (I, 2),(I, 3.2)                                                   5
obviously by_neqs so consider G such that between B D G by (II, 2)              6
consider E such that (∃a. on_line A a ∧ on_line D a ∧ on_line E a)               7
                                ∧ between C E G                                 8
    proof: clearly by_pasch consider E such that
        (∃a. on_line A a ∧ on_line D a ∧ on_line E a)
        ∧ (between B E C ∨ between C E G) by (II, 1) from 0,2,3,5,6
    obviously (by_eqs ∘ split) qed from 0,3,4,5 by (II, 1),(II, 3)
consider F such that (∃a. on_line C a ∧ on_line D a ∧ on_line F a)               9
                                ∧ between A F G                                 8
    [...]
have between A D E
    proof: obviously by_ncols so consider D' such that
        between C D' F ∧ between E D' A
        using K (MATCH_MP_TAC (5.3)) from 0,2,5,6,8,10 by (II, 1)
    obviously (by_eqs ∘ split) qed from 0,2,5,7,9 by (II, 1),(II, 3)
obviously by_ncols so consider B' such that
    between G D B' ∧ between C B' A
    using K (MATCH_MP_TAC (5.2)) from 0,2,5,7, by (II, 1)
obviously (by_eqs ∘ split) qed from 0,2,5,6 by (II, 1),(II, 3)
```

Figure 5.5: Verification of THEOREM 4

In these two applications of Axiom II, 4, we have used our `clearly` keyword to ask the `by_pasch` discoverer to search for a specific conclusion of the axiom. Exploiting our discoverer in this way makes things more robust than using `MATCH_MP_TAC` as we did in §5.1. With matching, we have to be careful that all the variables in the axiom line up with our desired conclusion. But when we use the discoverer, we know that its outputs are always lexicographically normalised, so there is only one conclusion we could possibly be after.

The sought conclusion could usually be copied and pasted from the many results obtained by the `by_pasch` discoverer when it was run concurrently during the interactive development, thereby selecting the intermediate result we want as a step in our verification and adding it to the proof context to be referenced later. Moreover, by using `clearly` and specifying the narrow set of justifying theorems needed to derive the conclusion, we make the pasted result the sole target for efficient search in proof replay.

The third and fourth applications of Pasch's axiom take the inner (5.3) and outer (5.2) forms respectively, and so we recommend the Outer Pasch Axiom as a useful lemma at this stage of Hilbert's exposition.

5.2.3 Alternative Proof

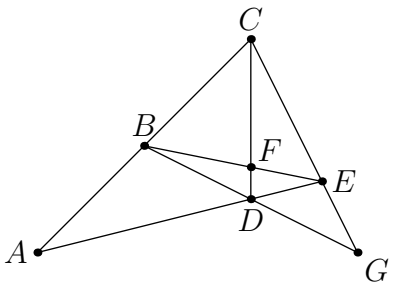
Our proof tool was mostly used in a supporting role, but for Theorem 4 we allowed it to tackle the problem unaided, probing into the search space by applying Axiom II, 4 non-deterministically. In this way, we hoped it would find alternative proofs of THEOREM 4, and, ideally, find one which required less than four applications of Axiom II, 4. This defined a search limit for the problem: once four applications were found, it stopped searching in the relevant branch.

We found several "alternatives", but most of these were symmetries of the original proof. In some cases, two independent applications of Pasch's axiom were applied in reverse order. In other cases, the proof was identical to the original up to a symmetric relabelling of the points. Only one new proof was revealed up to symmetry, and it corresponds to case (a) and (c) of Figure 5.4. We give it now in a prose formulation with an accompanying diagram.

Proposition. *Between points A and C is a third point B.*

Proof. Assume A, B and C are collinear, with A not between B or C and C not between A or B. We find a point D off the line AC and extend the segment BD to G using

Axiom II, 2. We then use Axiom II, 4 on the triangle BCG and the line AD to find the point E between C and G. We use Axiom II, 4 on the triangle BEG and the line CD to find the point F between B and E. We use the axiom again on the triangle ABE and the line CF to show that D lies between A and E. Finally, we can use the axiom on the triangle ACE and the line BG to find B between A and C. □



The opening strategy of our alternative proof is the same as the original. We first construct a point D off the line AC and extend the segment BD to G. This tells us that D lies between B and G, which gives us our first opportunities to use Axiom II, 4. In both cases, our goal is to use this axiom in order to place the point D between A and E, so that a final application of Pasch's axiom to the triangle ACE and the line BG will place B between A and C. The two proofs only differ in how they construct the point F, and how they use F to place D between A and E.

In Hilbert's proof, F is found on the edge of the outer triangle ACG, and is placed symmetrically with E. Indeed, the proof is valid even after exchanging all references of E and F, whereas in our proof, F is placed in the interior of ACG while E lies asymmetrically on the triangle's edge.

So Hilbert's proof has a lot of symmetry: E could be replaced with F; and the third application of Pasch's axiom could be made on the triangle CFG and the line AE, instead of AEG and the line CF. Our proof makes it clear that, while E and F can be constructed symmetrically and independently, only one of these points is distinguished in the final few steps.

It is worth drawing some attention to the subtlety of the incidence reasoning here. We could have applied Axiom II, 4 differently to find the point F, using the triangle CDG and the line BE. This would tell us that F lies on the line BE between C and D (before, it told us that F lies on the line CD between B and E). Now it might seem that we can use a symmetrical application of Pasch's axiom on the same line BE and the triangle ACD, which would solve the goal putting B between A and C. But at this stage in the proof, we must consider the possibility that BF exits the triangle ACD between A and

D. This possibility is not yet eliminable by incidence reasoning alone. It really does appear we need at least *four* applications of Axiom II, 4 to get this theorem.

Observations such as these are not apparent in Hilbert's proof. In his eleven uses of Axiom II, 4 across THEOREM 3, 4 and 5, Hilbert only considers the case-split implied by the axiom twice. And yet it takes up a significant amount of combinatorial reasoning about incidence. It is difficult to justify leaving this complexity implicit, when it has consequences on the shape of the proof which we find difficult to argue as obvious.

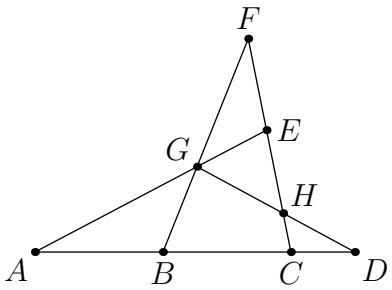
5.3 THEOREM 5

For the final theorem of this chapter, we shall see what is gained with the inner and outer form of Pasch's axiom, and we will look under the bonnet to see what our discoverers are actually up to.

THEOREM 5 has the most complex of the three proofs, taking up almost an entire page of the English translation. Like THEOREM 3 and THEOREM 4, it was originally an axiom in the first edition of Hilbert's text. The proof here is credited to E.H. Moore who proved it for projective geometry. Effectively, the result gives a transitivity property for point ordering. The proof is divided into three parts, though as observed by Dehlinger et al [21], it makes sense to leave the third part to the generalisation of THEOREM 6, which we cover in the next chapter.

5.3.1 Part 1 of THEOREM 5

> THEOREM 5. Given any four points on a line, it is always possible to label them A, B, C, D in such a way that the point labelled B lies between A and C and also between A and D, and furthermore, that the point labelled C lies between A and D and also between B and D.
>
> PROOF. Let A, B, C, D be four points on a line g. The following will now be shown:
>
> 1. If B lies on the segment AC and C lies on the segment BD then the points B and C also lie on the segment AD. By Axioms I, 3 and II, 2 choose a point E that does not lie on g, on [sic] a point F such that E lies between C and F. By repeated applications of Axioms II, 3 and II, 4 it follows that the segments AE and BF meet at a point G, and moreover, that the line CF meets the segment GD at a point H. Since H thus lies on the segment GD and since, however, by Axiom II, 3, E does not lie on the segment AG, the line EH by Axiom II, 4 meets the segment AD, i.e. C lies on the segment AD. In exactly the same way one shows analogously that B also lies on this segment.



[42, p. 7]

5.3.1.1 Evaluating our Manual Verification

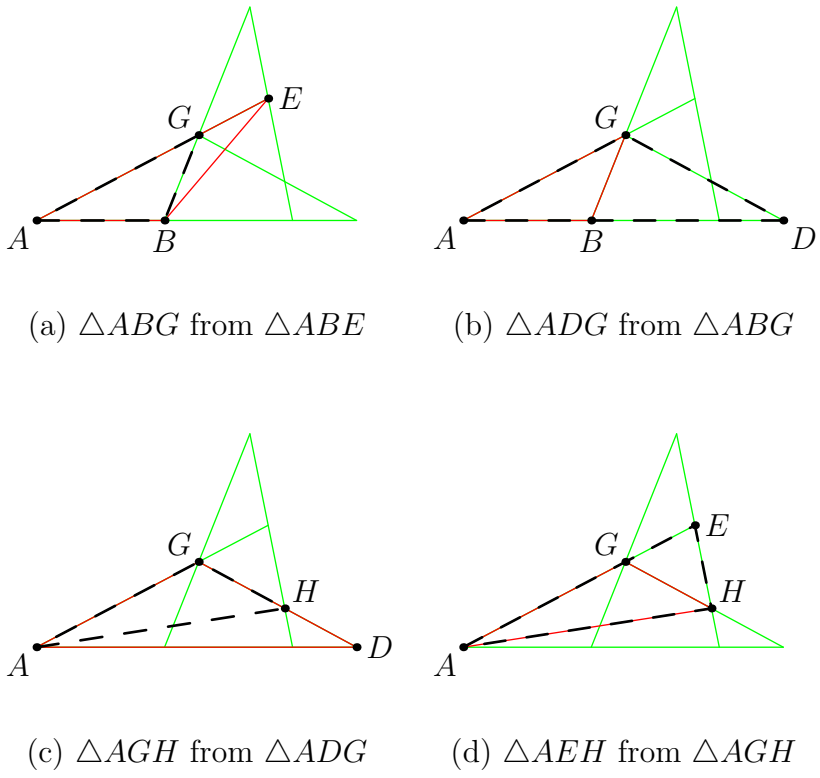
Our manual verification of this proof runs to approximately 80 lines of complicated proof steps. As we should expect by now, most of these steps were used to derive the preconditions needed for Axiom II, 4. These are now handled by our incidence discoverers.

In the manual verifications, the complexity of the inferences had got the better of us. We were not able to verify Hilbert's final application of Axiom II, 4 with the line EH and the triangle ADG. This requires knowing that the line EH does not intersect any vertex of ADG, which requires, in particular, knowing that AEH is a triangle.

We began to speculate that this matter was unprovable. In fact, we had produced a sketch argument that the derivation of $E \neq H$ was impossible, and we hoped that our discoverers would confirm this.

Instead, our discoverers refuted it. By tracking the path of inferences via a writer (see §4.7.2), we found that $\triangle AEH$ is derived at the end of a chain of discovered triangles starting from $\triangle ABE$. The rule linking each is `colncolncol` from §4.2.1, which can be understood as substituting points of a non-collinear triple one-at-a-time, until we have rewritten the initial triangle $\triangle ABE$ to $\triangle AEH$. We briefly discuss how.

It might seem that we can just replace the point B with the point H to rewrite $\triangle ABE$ to $\triangle AEH$, but the triangle introduction rule requires a hypothesis about an appropriate collinear set and an appropriate point inequality. Instead, the inference we use is less direct, and is shown in Figure 5.6. At first, our discoverer substitutes G for E using the line AGE, producing $\triangle ABG$ from $\triangle ABE$. It then substitutes D for B using the line ABD. This gives us a triangle ADG. Next, it substitutes H for D using the line DGH, giving us $\triangle AGH$. Finally, E and H are shown distinct on the basis of $\triangle AGH$ and the line AGE, after which the discoverer substitutes H for G using the line AGE, thus obtaining $\triangle AEH$.



(a) $\triangle ABG$ from $\triangle ABE$ (b) $\triangle ADG$ from $\triangle ABG$

(c) $\triangle AGH$ from $\triangle ADG$ (d) $\triangle AEH$ from $\triangle AGH$

Figure 5.6: Finding $\triangle AEH$

5.3.1.2 Strengthening Inner and Outer Pasch

We have a verification of THEOREM 5 using Axiom II, 4 directly via `by_pasch`. Incidence reasoning implicitly dominates this verification, but our discoverers take on the labour. We can avoid much of this implicit reasoning though, and so in another verification, we exploit the Inner and Outer Pasch axioms, whose preconditions have only one incidence assumption: namely that the triangle on which the axiom applies exists. By using this form of the axiom, a much cleaner proof can be obtained, one which needs much less automation. Furthermore, we do not need to write any subproofs to identify points in the figure, nor eliminate disjuncts.

Our versions of Inner and Outer Pasch are strengthened from their axiomatic form. Consider Veblen's Outer Pasch Axiom, which we can formalise as

$$\begin{aligned}\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)& \\ \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ C& \\ \Longrightarrow\ \exists F.\ \exists a.\ \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{between}\ A\ F\ B.&\end{aligned}$$

We can say something stronger in the conclusion here. We know that D, E and F

are not merely collinear. The point E must lie between D and F (see the diagram accompanying Veblen's proof in §5.1.2). This is an important corollary, as evidenced by the fact that it is the very first theorem Veblen proves after stating the axiom. Thus, we have both Inner and Outer Pasch axioms as the following strengthened theorems:

$$\begin{aligned} \vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ C \\ \Longrightarrow \exists F.\ \texttt{between}\ D\ E\ F \wedge \texttt{between}\ A\ F\ B. \end{aligned} \tag{5.2}$$

$$\begin{aligned} \vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ B \\ \Longrightarrow \exists F.\ \texttt{between}\ D\ F\ E \wedge \texttt{between}\ A\ F\ C. \end{aligned} \tag{5.3}$$

```
assume between A B C ∧ between B C D                                          0, 1
consider E such that ¬(∃a. on_line A a ∧ on_line B a ∧ on_line E a)
    from 0 by (I, 2), (I, 3.2), (II, 1)                                       2
obviously by_neqs so consider F such that
    between C E F from 0 by (II, 2)                                           3
obviously by_ncols so consider G such that
    between A G E ∧ between B G F from 0, 2, 3 by (II, 1), (5.3)              4, 5
obviously by_ncols so consider H such that
    between C H F ∧ between D H G from 0, 1, 2, 3, 5 by (II, 1), (5.3)        6
have A ≠ D from 0, 1 by (II, 1), (II, 3)
obviously by_ncols so consider C' such that
    between E H C' ∧ between A C' D from 0, 1, 2, 4, 6, 7 by (5.2), (II, 1)
obviously (by_eqs ∘ split) qed from 0, 1, 2, 3, 6, 7
```

Figure 5.7: THEOREM 5 verification, part 1

5.3.1.3 Verification

With these theorems now derived and with some of our automation, we obtain the verification in Figure 5.7, which is very close to the prose. We have two steps which

Hilbert does not make explicit. Firstly, we note that $A \neq D$, a fact which follows from our assumptions and Axiom II, 3, and without which we would not be able to show the existence of $\triangle ADG$ for our final application of Pasch's axiom.

The other extra step is somewhat of an irritation. Instead of applying Pasch's axiom to conclude that C is between A and D, we must instead obtain a new point C' and then use incidence reasoning via `by_eqs` to identify it with C.

5.3.1.4 Comparison with the Prose

The strengthened versions of the Pasch axioms mean we can be more efficient than Hilbert, who uses an unspecified number of applications of Axiom II, 4.

> "By repeated applications of Axioms II, 3 and II, 4 it follows that the segments AE and BF meet at a point G, and moreover, that the line CF meets the segment GD at a point H."
>
> [42, p. 7]

We can say exactly how many applications are needed here. In one of our verifications of THEOREM 5, which avoids the Inner and Outer Pasch axioms and so follows Hilbert most closely, we apply Axiom II, 4 via the `by_pasch` discoverer. By doing so, we see there are exactly three applications implied by Hilbert's prose (we elide the subproofs used to reject offending disjuncts).

```
consider G such that (∃a. on_line B a ∧ on_line F a ∧ on_line G a)
                     ∧ between A G E                                  4,5
    proof: clearly by_pasch ...
have between B G F                                                    6
    proof: clearly by_pasch ...
consider H such that (∃a. on_line C a ∧ on_line F a ∧ on_line H a)
                     ∧ between D H G                                  7,8
    proof: clearly by_pasch ...
```

That three applications are necessary is implied by the careful language used in the prose: "the segments AE and BF meet at a point G" while "the *line* CF meets the segment GD at a point H" (our emphasis). Now if we are to show that two *segments* intersect, we must derive *two* facts of betweenness, and therefore we need *two* applications of Axiom II, 4. But if we are to show that a *line* and a segment intersect, we need only one fact of betweenness.

In our verification using the Inner and Outer Pasch axioms (Figure 5.7), we can trim this down. The intersection of the segments AE and BF is covered by just one application of the strengthened version of the Inner Pasch Axiom (5.3), which we use again to intersect the segments CF and GD. Finally, we use the Outer Pasch Axiom (5.2) to find a point C' between A and D.

For this final application of Pasch's axiom, Hilbert writes: "...and since, however, by Axiom II, 3, E does not lie on the segment AG,...." We draw attention to this remark because it is not reflected in our verification. Hilbert, for the one and only time, is explicitly eliminating the disjunct in the conclusion of Axiom II, 4, by appealing to the fact that G lies between A and E. We must appeal to the same fact, but in our verification, it is just the necessary precondition of the inner Pasch axiom (5.3).

Finally, we mention Hilbert's final step "one shows analogously that B also lies on this segment." This does not require an analogous *proof* as the word "show" would imply. Instead, we can capture the analogy directly by using the theorem verified in Figure 5.7 as a lemma and then applying the symmetry of betweenness. In fact, `MESON` takes care of this automatically:

```
MESON [lemma,(II, 1)] ∀A. ∀B. ∀C. ∀D. between A B C ∧ between B C D
                                    ⟹ between A B D ∧ between A C D
```

5.3.2 Discovery at work

In this subsection, we give an idea of how our discoverers interact with HOL Light by showing the sequents generated concurrently as we interactively develop a declarative verification of THEOREM 5. We consider two discoverers, `by_incidence`, which generates the five kinds of basic incidence sequents described in the last chapter, and `by_pasch`, which finds potential applications of Pasch's axiom. The `by_pasch` discoverer feeds off the generations produced by `by_incidence`, so that their work is not duplicated. But there is no *feedback*. That is, `by_incidence` does not continue searching based on the results of `by_pasch`. Instead, we, the user, shall take responsibility for when Pasch's axiom is applied. It is a point introduction axiom explicit in the prose that we want to keep explicit in the verification.

The discovery begins once we have stated the theorem's assumptions and obtained our first non-collinear point.

`assume between` $A\ B\ C \wedge$ `between` $B\ C\ D$	0, 1
`consider` E `such that` $\neg(\exists a.$ `on_line` $A\ a \wedge$ `on_line` $B\ a \wedge$ `on_line` $E\ a)$	
`from` 0 `by` (I, 2), (I, 3.2), (II, 1)	2

Concurrently, sequents are pulled from the discoverer `by_incidence`, which lazily forces values according to our data-flow diagram from Figure 4.1. This ultimately involves pulling hypothesis sequents from the discoverer `monitor`, whose job it is to inspect the proof context constructed from the steps of the declarative proof, as it is written, and add any new hypotheses which appear there. Here, we have three hypotheses, which are picked up and fed through our incidence discoverers to produce the seven generations of incidence sequents shown in Figure 5.8. If the stream is forced beyond this, only empty generations appear, indicating that no more inference is possible.

The seven generations of sequents are delivered within 0.31 seconds.[2]. We write the dependent hypotheses in parentheses, and omit the turnstile ($\vdash$) and sequent context. Note that sequents can be repeated if they can be derived in more than one way. When we are not tracking dependent hypotheses, such repetitions are automatically filtered out.

As we can see, two of the seven generations are empty. This happens because of filtering: some of the inferences we use turn out to generate sequents which have already appeared, and duplicates are always removed from the discoverer. This filtering sometimes leaves generations completely empty.

Besides the selection of triangles here, we have a sequent which says that all points in our figure lie in the same plane. We can see how this sequent has grown over the generations, with larger and larger planar sets found, via rule `planeplane` from §4.2.1.

To follow Hilbert's proof at this stage, all we need to know is that $C \neq E$, a fact which is delivered in the fourth generation. We are told that its derivation depends on hypothesis 0, namely `between` $A\ B\ C$, and hypothesis 2, which is

$$\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a).$$

Since this was the *last* hypothesis to enter the proof context, we can add it as justification to the declarative proof we are building by using the Mizar Light keyword `so`. Again, appealing to the pertinent hypotheses gives the reader more information

[2]We have tested this on an Intel Core 2 with a 2.53GHz clock speed.

about the dependencies within the proof, and enables the discoverer to work more efficiently in replay, where it will use only those hypotheses that have been marked as justification.

```
obviously by_neqs so consider F such that between C E F from 0 by (II, 2)   3
```

$$
\left\{
\begin{array}{ll}
\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a, & (1)\\
\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a, & (0)\\
\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a), & (2)\\
B \neq C, B \neq D, C \neq D, & (1)\\
A \neq B, A \neq C, B \neq C, & (0)\\
A \neq B, A \neq E, B \neq E, & (2)\\
\exists \alpha.\ \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha, & (1)\\
\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha, & (0)\\
\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha & (2)
\end{array}
\right\},
$$

$$\{\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha \quad (0,1)\},$$

$$\{\},$$

$$
\left\{
\begin{array}{ll}
\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a), & (0,2)\\
\neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a), & (0,2)\\
C \neq E, & (0,2)\\
\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha & (0,2)
\end{array}
\right\},
$$

$$\{\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \quad (0,1)\},$$

$$\{\},$$

$$
\left\{
\begin{array}{ll}
\neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a), & (0,1,2)\\
\neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a), & (0,1,2)\\
D \neq E, & (0,1,2)\\
\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha & (0,1,2)
\end{array}
\right\}
$$

Figure 5.8: First Generations of Discovered Sequents

With this declarative proof step, we add a new sequent into the proof-context, which is picked up by the `monitor` discoverer, to flow into the network of incidence discoverers, creating new generations of sequents. These generations are shown in Figure 5.9 and are found within 1.21 seconds. Including the sequents found earlier, we have thirteen triangles identified in total, and so we are going to be interested in which applications of Axiom II, 4 are permissible. To find out, we look more specifically at the results

of our `by_pasch` discoverer. Its first eighteen generations are empty, meaning that it requires a search-depth of eighteen before all the required preconditions of Pasch have been found. We are then told that, in this early stage of the proof, there are already six possibilities to choose from. The full set is shown in Figure 5.10 and is found within 2.82 seconds.

$$
\left\{\begin{array}{ll}\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a, & (3)\\ C \neq E, C \neq F, E \neq F, & (3)\\ \exists \alpha.\ \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha & (3)\end{array}\right\},
$$

$$
\left\{\begin{array}{ll}\exists \alpha.\ \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha, & (1,3)\\ \exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha & (0,3)\end{array}\right\},
$$

$$
\left\{\begin{array}{ll}\exists \alpha.\ \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha, & (1,3)\\ \exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha & (0,3)\end{array}\right\},
$$

$$
\{\},\{\},
$$

$$
\left\{\begin{array}{ll}\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha & \\ \qquad \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha \wedge \texttt{on\_plane}\ F\ \alpha & (0,1,3)\end{array}\right\},
$$

$$
\{\},
$$

$$
\{A \neq F, B \neq F \quad (0,2,3)\},
$$

$$
\{\},\{\},
$$

$$
\{D \neq F \quad (0,1,2,3)\},
$$

$$
\{\},\{\},\{\},
$$

$$
\left\{\begin{array}{ll}\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ F\ a), & (0,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a), & (0,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ F\ a), & (0,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a) & (0,2,3)\end{array}\right\},
$$

$$
\{\},\{\},
$$

$$
\left\{\begin{array}{ll}\neg(\exists a.\ \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a), & (0,1,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ F\ a), & (0,1,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ F\ a), & (0,1,2,3)\\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ F\ a) & (0,2,3)\end{array}\right\}
$$

Figure 5.9: Second generations of sequents

The last possibility corresponds to one of Hilbert's applications of Axiom II, 4, which we shall therefore add as an explicit proof step in the declarative proof we are building.

$$\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},$$
$$\left\{\begin{array}{ll} \exists G.\,(\exists a.\,\texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ A\ G\ C \vee \texttt{between}\ A\ G\ F), & (0,2,3) \\ \exists G.\,(\exists a.\,\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ B\ G\ C \vee \texttt{between}\ B\ G\ F) & (0,2,3) \end{array}\right\},$$
$$\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},$$
$$\left\{\begin{array}{ll} \exists G.\,(\exists a.\,\texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ B\ G\ C \vee \texttt{between}\ B\ G\ F), & (0,1,2,3) \\ \exists G.\,(\exists a.\,\texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ C\ G\ D \vee \texttt{between}\ D\ G\ F), & (0,1,2,3) \\ \exists G.\,(\exists a.\,\texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ A\ G\ F \vee \texttt{between}\ C\ G\ F) & (0,2,3) \end{array}\right\},$$
$$\{\},\{\},\{\},\{\},\{\},$$
$$\left\{\begin{array}{ll} \exists G.\,(\exists a.\,\texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a) & \\ \qquad \wedge (\texttt{between}\ A\ G\ E \vee \texttt{between}\ C\ G\ E) & (0,2,3) \end{array}\right\}$$

Figure 5.10: Discovered applications of Pasch's Axiom

We do this by asserting the theorem as `clearly` derivable, adding the three hypotheses as explanatory justification, so that the theorem is found efficiently in replay:

```
clearly by_pasch so consider G such that
(∃a. on_line B a ∧ on_line F a ∧ on_line G a) ∧ (between A G E ∨ between C G E)
    from 0,2
```

We now have a disjunction in our hypotheses. When we run the `by_incidence` discoverer with the `split` function, it will convert this disjunction into a tree, which will combine as described in §4.5.1. Our generations become proper trees, and the search space is automatically partitioned into three.

The first partition covers inferences which are made in the root node of our trees, where no particular disjunct is assumed. These generations are shown in Figure 5.11. Here, the discoverer infers that the new point G must, *in any case*, be distinct from A, C, D and E, and that all seven points must be planar.

$$\{\texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a \quad (4)\},$$
$$\{\},\{\},\{\},\{\},\{\},\{\},$$
$$\left\{\begin{array}{ll} \exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ D\ \alpha & \\ \qquad \wedge \texttt{on\_plane}\ E\ \alpha\ \texttt{on\_plane}\ F\ \alpha\ \texttt{on\_plane}\ G\ \alpha & (0,1,3,4) \end{array}\right\},$$
$$\{\},$$
$$\{C \neq G, E \neq G \quad (0,2,3,4)\},$$
$$\{\},\{\},$$
$$\left\{\begin{array}{ll} D \neq G, & (0,1,2,3,4), \\ A \neq G & (0,2,3,4) \end{array}\right\}$$

Figure 5.11: Generations in the root of the case-split

$$\left\{\begin{array}{ll} \exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a & (4), \\ A \neq G, A \neq E, E \neq G & (4) \end{array}\right\},$$
$$\{\},\{\},$$
$$\{B \neq G \quad (2,4)\}$$
$$\{\},\{\},$$
$$\{C \neq G \quad (0,2,4)\}$$
$$\{F \neq G \quad (0,2,3,4)\}$$
$$\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},\{\},$$
$$\left\{\begin{array}{ll} \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ G\ a) & (2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & (2,4) \end{array}\right\},$$
$$\{\},\{\},\{\}$$
$$\left\{\begin{array}{ll} \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \end{array}\right\},$$
$$\{\},\{\},$$
$$\left\{\begin{array}{ll} \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,3,4) \\ \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,3,4) \\ \neg(\exists a.\ \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,3,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ G\ a) & (0,1,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ G\ a) & (0,1,2,4) \\ D \neq G & (0,1,2,4) \end{array}\right\}.$$

Figure 5.12: Generations found on the assumption `between` $A\ G\ E$

We get more information in the next partition, which is the left-branch of the case-split, shown in Figure 5.12. When the trees are flattened, the sequents in this branch carry `between` $A\ G\ E$ as a disjunctive hypothesis. For brevity, we omit sequents about the existence of planes.

This is the consistent case. The triangles found here will be used as further justification for applications of Axiom II, 4. The inconsistent case occurs in the remaining partition, carrying the disjunctive hypothesis `between` $C\ G\ E$. See Figure 5.13.

$$\left\{\begin{array}{ll} \exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a & (4), \\ C \neq G, C \neq E, E \neq G & (4) \end{array}\right\},$$

$$\{\},\{\},\{\},$$

$$\{\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a \wedge \texttt{on\_line}\ G\ a, \quad (3,4)\},$$

$$\{\}$$

$$\{A \neq G, B \neq G \quad (0,2,4)\},\{\},\{\},\{D \neq G \quad (0,1,2,4)\}$$

$$\{\},$$

$$\left\{\begin{array}{ll} \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \end{array}\right\},\{\},\{\},$$

$$\left\{\begin{array}{ll} \neg(\exists a.\ \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ G\ a) & (0,1,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ G\ a) & (0,1,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ G\ a) & (0,1,2,4) \\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ G\ a) & (0,2,4) \\ F = G & (0,2,3,4) \end{array}\right\}$$

Figure 5.13: Generations found on the assumption `between` $C\ G\ E$

The use of our tree data-structure means this last sequence is generated in parallel with the other two. It takes 9.58 seconds to generate all three sequences, though most of the theorems which appear are generated in under 1 second. The theorem required to advance the proof

$$\texttt{between}\ C\ G\ E \implies F = G$$

is generated in 6.19 seconds. The consequent of this implication contradicts our hypothesis that `between` $C\ E\ F$ via (II, 3), and thus, the case given by its antecedent can be discarded.

```
obviously (by_eqs ∘ split) qed by (II, 1), (II, 3) from 0, 2, 3
```

The `obviously` keyword tells the step to collapse the stream of trees, pushing the branch labels into the sequents as antecedents. It then uses the resulting discovered sequents to justify the step.

The rest of the proof proceeds similarly. With the help of our discoverers, we reduce the 80 or so steps used in our manual verification to a verification with just 17 steps. Generally, we found a consistent reduction of roughly 80% in proof length across all 18 theorems from our manual verifications, with the new verifications always comparing much more favourably with the prose.

5.3.3 Part 2 of THEOREM 5

We now consider the second part of THEOREM 5, our verification of which matches Hilbert's logic very closely, even if we have reordered some of the derivations. The whole proof is indirect, which explains why there is no accompanying diagram, and it is one of the few proofs where Hilbert treats the disjunction in Axiom II, 4 symmetrically: both alternatives entail a contradiction.

> 2. If B lies on the segment AC and C lies on the segment AD then C also lies on the segment BD and B also lies on the segment AD. Choose one point G that does not lie on g, and another point F such that G lies on the segment BF. By Axioms I, 2 and II, 3 the line CF meets neither the segment AB nor the segment BG and hence, by Axiom II, 4 again, does not meet the segment AG. But since C lies on the segment AD, the straight line CF meets then the segment GD at a point H. Now by Axiom II, 3 and II, 4 again the line FH meets the segment BD. Hence C lies on the segment BD. The rest of Assertion 2 thus follows from 1.
>
> [42, p. 7]

Our verification is shown in Figure 5.14. In the prose, Hilbert is applying Axiom II, 4 in its contrapositive form, so we cannot follow him literally by using our `by_pasch` discoverer. Instead, we run a *reductio* argument in a subproof, where we can use Axiom II, 4 in a forward direction.

Notice that this is the one time that Hilbert makes an explicit reference to an incidence axiom other than I, 3: he cites Axiom I, 2, and we do not have a clear idea why. This axiom is needed in many places in these proofs, but is nearly always implicit. Bernays cites the same axiom in a proof supplementing the text (see §7.2.2). The only obvious

commonality between the two is that the citation occurs when Axiom II, 4 is applied and *both* disjuncts in the conclusion are eliminated.

But Axiom I,2 is required when eliminating even *one* disjunct. So perhaps there is evidence here that neither Hilbert nor Bernays are taking the case-split in Axiom II, 4 sufficiently seriously. If so, then it makes sense for our verifications to eliminate the disjuncts explicitly in subproofs, and it makes sense that we treat Axiom I, 2 uniformly, and leave its use implicit.

```
assume between A B C ∧ between A C D                                          0,1
consider G such that
    ¬(∃a. on_line A a ∧ on_line B a ∧ on_line G a) from 0 by (I, 2),(I, 3.2),(II, 1)   2
obviously by_neqs so consider F such that between B G F by (II, 2)            3
have ¬(∃P. (∃a. on_line C a ∧ on_line F a ∧ on_line P a) ∧ between A P G       4
    proof: otherwise consider P such that
        ∃P. (∃a. on_line C a ∧ on_line F a ∧ on_line P a) ∧ between A P G      4,5
    clearly by_pasch so consider Q such that
        (∃a. on_line C a ∧ on_line P a ∧ on_line Q a)
            (between A Q B ∨ between B Q G) from 0,2
    obviously (by_eqs ∘ split) qed from 0,2,3,4,5 by (II, 1),(II, 3)
obviously by_pasch so consider H such that
    (∃a. on_line C a ∧ on_line F a ∧ on_line H a)
        ∧ between D H G from 0,1,2,3 at 5,6
have B ≠ D from 0,1 by (II, 1),(II, 3)                                        7
clearly by_pasch so consider C′ such that
    (∃a. on_line C′ a ∧ on_line F a ∧ on_line H a)
        (between B C′ D ∨ between B C′ G from 0,1,2,3,5,6
obviously (by_eqs ∘ split) qed from 0,1,2,3,5,6,7 by (II, 1),(II, 3)
```

Figure 5.14: THEOREM 5 verification, part 2

5.4 Conclusion

Hilbert gives only three prose proofs for his first two groups in the *Grundlagen der Geometrie*. None of these proofs were present in the first edition, and the results they prove, one-dimensional ordering theorems, were originally assumed as axioms. Two of the proofs were contributed by Wald and E.H. Moore, though Veblen also deserves credit in helping to investigate how linear order theorems can be derived from two-dimensional order axioms.

We have reviewed how the automation we discussed in the last chapter enables us to write very short proofs compared to our manual verifications, without having to break from Hilbert's basic proof strategies. In fact, we obtain proofs whose steps match Hilbert's prose steps very closely. Our DeBruijn factor is almost 1, and our incidence discoverers appear to be adequate to handle all of the incidence reasoning implicit in Hilbert's proofs.

This is not merely aesthetic. That we can write such short verifications of these relatively simple theorems without trudging through a bog of incidence arguments gave us hope that we could tackle the far more complex verification of the Polygonal Jordan Curve Theorem. Indeed, we shall find that the automation described in the last chapter is used aggressively in that verification.

We just need some further automation for dealing with linear ordering. This is described in the next chapter, which shows how THEOREM 4 and THEOREM 5 are generalised in THEOREM 6.

Chapter 6

Infinity and Linear Ordering

The next theorem on the agenda is THEOREM 6, and in this chapter, we will look at two different ways to deal with it formally. The theorem itself tells us that any finite set of points on a line is linearly ordered in terms of betweenness. One approach to formalising this is at the meta-level, where it can be treated as an algorithm for enumerating cases of betweenness. In another approach, we can formalise the theorem at the object level and verify it. In so doing, we shall derive the axiom of infinity, thereby showing the axiom redundant given Hilbert's geometric axioms.

6.1 THEOREM 6 at the Meta-level

We ended the last chapter by discussing two of the three parts of Hilbert's proof of THEOREM 5. The third and final part of the proof can be generalised to verify THEOREM 6.

> THEOREM 6 (generalisation of THEOREM 5). Given any finite number of points on a line it is always possible to label them A, B, C, $D, \ldots, K$ in such a way that the point labelled B lies between A, and C, D, $E, \ldots, K$, the point labelled C lies between A, B and D, $E, \ldots K$, D lies between A, B, C and $E, \ldots$, etc. Besides this order of labelling there is only the reverse one that has the same property.
>
> [42, pp. 7-8]

It strikes us that a *labelling* of points is a syntactic device, a reference to the symbols used to mention points rather than a reference to the points themselves. Thinking more formally, we could identify a labelling with an assignment of values to variables.

The existence of a labelling is then the existence of these assignments. With this in mind, a first pass at trying to faithfully formalise THEOREM 6 could view it as a *schema*, asserting that, given some points on a line A', B', C', D', …, K', there exists an appropriate assignment to the points A, B, C, D, …, K:

$$\begin{aligned}
&\exists a.\ \texttt{on_line}\ A'\ a \wedge \texttt{on_line}\ B'\ a \wedge \texttt{on_line}\ C'\ a \wedge \texttt{on_line}\ D'\ a \\
&\qquad \wedge \texttt{on_line}\ E'\ a \wedge \cdots \wedge \texttt{on_line}\ K'\ a \\
&\implies \exists A.\ \exists B.\ \exists C.\ \exists D.\ \ldots\ \exists K. \\
&\qquad A = A' \wedge B = B' \wedge C = C' \wedge D = D' \wedge E = E' \wedge \cdots \wedge K = K' \\
&\qquad \wedge \begin{pmatrix} \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ B\ D \wedge \texttt{between}\ A\ B\ E \\ \qquad \wedge \cdots \wedge \texttt{between}\ A\ B\ K \end{pmatrix} \\
&\qquad \wedge \begin{pmatrix} \texttt{between}\ A\ C\ D \wedge \texttt{between}\ A\ C\ E \wedge \cdots \wedge \texttt{between}\ A\ C\ K \\ \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ B\ C\ E \wedge \texttt{between}\ B\ C\ K \\ \qquad \wedge \cdots \wedge \texttt{between}\ B\ C\ K \end{pmatrix} \\
&\qquad \wedge \begin{pmatrix} \texttt{between}\ A\ D\ E \wedge \cdots \wedge \texttt{between}\ A\ D\ K \\ \wedge \texttt{between}\ B\ D\ E \wedge \cdots \wedge \texttt{between}\ B\ D\ K \\ \wedge \texttt{between}\ C\ D\ E \wedge \cdots \wedge \texttt{between}\ B\ D\ K \end{pmatrix} \\
&\qquad \vdots
\end{aligned}$$

This is not a formula of higher-order logic. Instead, as a schema, it tells us that as we fix our choice of variables $A', B', C', D', E', \ldots, K'$, we are expected to fill in the holes by continuing a syntactic pattern.

Alternatively, we could say this is a metatheorem, something surprisingly typical of mathematics. As noted by Harrison [34], the existence of such theorems must be borne in mind when formalising, and he mentions the simple example of an object theorem about certain operations being associative and commutative (say, addition), which entails the *metatheorem* that brackets (bits of syntax) can be dropped without ambiguity.

Hilbert was working before any distinction had been made between meta-level and object level, let alone the sort of hard distinction we now have between HOL and the computational metalanguage ML which encodes its syntax and inference rules. With such a distinction, we immediately realise that our schema above can be formalised as an ML function which inputs a list of points, instantiates the schema to create a HOL formula, and then derives it as a theorem.

$$\begin{aligned}
&\texttt{between}\ A'\ B'\ C' \wedge \texttt{between}\ A'\ B'\ D' \wedge \texttt{between}\ A'\ C'\ D' \wedge \texttt{between}\ B'\ C'\ D' \\
&\vee \texttt{between}\ A'\ B'\ D' \wedge \texttt{between}\ A'\ B'\ C' \wedge \texttt{between}\ A'\ D'\ C' \wedge \texttt{between}\ B'\ D'\ C' \\
&\vee \texttt{between}\ A'\ C'\ B' \wedge \texttt{between}\ A'\ C'\ D' \wedge \texttt{between}\ A'\ B'\ D' \wedge \texttt{between}\ C'\ B'\ D' \\
&\vee \texttt{between}\ A'\ C'\ D' \wedge \texttt{between}\ A'\ C'\ B' \wedge \texttt{between}\ A'\ D'\ B' \wedge \texttt{between}\ C'\ D'\ B' \\
&\vee \texttt{between}\ A'\ D'\ B' \wedge \texttt{between}\ A'\ D'\ C' \wedge \texttt{between}\ A'\ B'\ C' \wedge \texttt{between}\ D'\ B'\ C' \\
&\vee \texttt{between}\ A'\ D'\ C' \wedge \texttt{between}\ A'\ D'\ B' \wedge \texttt{between}\ A'\ C'\ B' \wedge \texttt{between}\ D'\ C'\ B' \\
&\vee \texttt{between}\ B'\ A'\ C' \wedge \texttt{between}\ B'\ A'\ D' \wedge \texttt{between}\ B'\ C'\ D' \wedge \texttt{between}\ A'\ C'\ D' \\
&\vee \texttt{between}\ B'\ A'\ D' \wedge \texttt{between}\ B'\ A'\ C' \wedge \texttt{between}\ B'\ D'\ C' \wedge \texttt{between}\ A'\ D'\ C' \\
&\vee \texttt{between}\ B'\ C'\ A' \wedge \texttt{between}\ B'\ C'\ D' \wedge \texttt{between}\ B'\ A'\ D' \wedge \texttt{between}\ C'\ A'\ D' \\
&\vee \texttt{between}\ B'\ D'\ A' \wedge \texttt{between}\ B'\ D'\ C' \wedge \texttt{between}\ B'\ A'\ C' \wedge \texttt{between}\ D'\ A'\ C' \\
&\vee \texttt{between}\ C'\ A'\ B' \wedge \texttt{between}\ C'\ A'\ D' \wedge \texttt{between}\ C'\ B'\ D' \wedge \texttt{between}\ A'\ B'\ D' \\
&\vee \texttt{between}\ C'\ B'\ A' \wedge \texttt{between}\ C'\ B'\ D' \wedge \texttt{between}\ C'\ A'\ D' \wedge \texttt{between}\ B'\ A'\ D'.
\end{aligned}$$

Figure 6.1: THEOREM 5 case-split

6.1.1 Representation

Consider applying THEOREM 6 to the special case of THEOREM 5 where there are only four points A', B', C', D' and four labels A, B, C and D. The formula we must verify is:

$$\begin{aligned}
&\exists a.\ \texttt{on_line}\ A'\ a \wedge \texttt{on_line}\ B'\ a \wedge \texttt{on_line}\ C'\ a \wedge \texttt{on_line}\ D'\ a \\
&\quad\Longrightarrow\ \exists A.\ \exists B.\ \exists C.\ \exists D.\ A = A' \wedge B = B' \wedge C = C' \wedge D = D' \\
&\qquad\qquad \wedge \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ B\ D \wedge \texttt{between}\ A\ C\ D \wedge \texttt{between}\ B\ C\ D.
\end{aligned}$$

Now we can "unwind" this existential into the disjunction given in Figure 6.1, and thus obtain a concrete formalisation of THEOREM 5. It is rather verbose though, and as we increase the number of points, the number of disjuncts will explode.

Hilbert does not state THEOREM 6 with any efficiency in mind, as he lists all possible betweenness relations among the points considered. We shall not be so wasteful, and will control the blow-up by finding a concise, canonical representation of the linear order of points on a line. The constraint is that we should be able to quickly derive any member of the full set of betweenness relations from the canonical representation.

Firstly, we must note the fact that Hilbert's geometry has no preferred orientation. For instance, there is no preferred x, y and z direction and no preferred clockwise or anticlockwise direction for three non-collinear points. This is in contrast to coordinate free methods such as the signed-area method [46]. But in Hilbert's geometry, we *can* treat the first argument to the between predicate as a parameter giving the preferred origin. Then, we can regard the partially applied relation as an ordering on every possible ray emanating from the origin. More precisely, we can read $\texttt{between}\ A\ B\ C$ as saying that $B < C$ from the perspective of origin A and direction $\overrightarrow{AB}$.

We can then represent total orders as conjunctions ordering adjacent points. So, if A,B,C,D,E,F,G,H occur along a line in that order, we just need the six conjuncts

$$\Gamma \vdash \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \wedge \texttt{between}\ A\ D\ E \\ \wedge \texttt{between}\ A\ E\ F \wedge \texttt{between}\ A\ F\ G \wedge \texttt{between}\ A\ G\ H. \quad (6.1)$$

We want to retrieve all other betweenness relations implied by this sequent quickly. Before we describe how we do this, we will separate THEOREM 5, whose verification we considered in the last chapter, into two separate theorems. The first theorem (6.2) will "move the origin" from A to B in our representation. The second theorem (6.3), when we understand the `between` relation as a binary relation whose first argument is the origin parameter, gives us transitivity.

$$\vdash \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \implies \texttt{between}\ B\ C\ D. \quad (6.2)$$

$$\vdash \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \implies \texttt{between}\ A\ B\ D. \quad (6.3)$$

We now explain our strategy by way of example. Suppose our goal is to derive $\texttt{between}\ C\ F\ H$ from (6.1). Since the ordering in (6.1) has origin A and our goal has origin C, our first task is to move the origin from C to A. To do so, we match the goal against the conclusion of Theorem 6.2, giving the obligation

$$\texttt{between}\ A\ C\ F \wedge \texttt{between}\ A\ F\ H.$$

Interpreting `between` as a binary relation, we are obliged to show that $C < F$ and $F < H$ with respect to origin A and direction $\overrightarrow{AC}$. The bounds here cover a range of points from C to H, and so we split the ordering of (6.1) from C to H into two suborderings either side of F. That is, we extract the two conjunctions

$$\texttt{between}\ A\ C\ D \wedge \texttt{between}\ A\ D\ E \wedge \texttt{between}\ A\ E\ F$$

and

$$\texttt{between}\ A\ F\ G \wedge \texttt{between}\ A\ G\ H.$$

We now reason transitively, obtaining $\texttt{between}\ A\ C\ F$ and $\texttt{between}\ A\ F\ H$ by folding (6.3) across their conjuncts:

$$\begin{aligned}\texttt{between}\ A\ C\ D \wedge \texttt{between}\ A\ D\ E \wedge \texttt{between}\ A\ E\ F \\ \longrightarrow \texttt{between}\ A\ C\ E \wedge \texttt{between}\ A\ E\ F \\ \longrightarrow \texttt{between}\ A\ C\ F\end{aligned}$$

and

$$\begin{aligned}\texttt{between}\ A\ F\ G \wedge \texttt{between}\ A\ G\ H \\ \longrightarrow \texttt{between}\ A\ F\ H.\end{aligned}$$

The example here generalises and we have implemented it as a completely deterministic tactic, taking a sequent such as (6.1) and a formula such as $\texttt{between}\ C\ F\ H$, and then verifying the formula in one pass of the order conjunction.

6.1.2 Enumerating Possible Orderings

Even with the more concise representation, there are $\frac{1}{2}n!$ possible orderings to consider for n points, though in practice, there are constraints on the ordering that allow us to eliminate many of the cases by appealing to Axiom II, 3. For instance, if we know that $\texttt{between}\ B\ A\ C$ and $\texttt{between}\ B\ D\ C$, then there are only two ways to order A, B, C and D:

$$\begin{aligned}\texttt{between}\ B\ A\ D \wedge \texttt{between}\ B\ D\ C \\ \vee \texttt{between}\ B\ D\ A \wedge \texttt{between}\ B\ A\ C.\end{aligned}$$

Factoring in these constraints not only cuts down the size of the final conclusion, but significantly speeds up the calculation of the possibilities. We have thus implemented a procedure to enumerate all possible orderings, taking both a list of the points we want it to order, and a list of betweenness sequents constraining the possibilities.

The algorithm nicely captures the purpose of a metatheorem such as Hilbert's if we see it as a computation on labellings. It also keeps us well within the scope of first-order logic. However, our ML algorithm is only a verification of the particular instances it generates. The algorithm, the metatheorem itself, is unverified. To verify it, we must bring the meta-level down to the object level.

6.2 THEOREM 6 at the Object Level

Any logic which can formalise THEOREM 6 as a theorem rather than a metatheorem will need to include a domain of labellings. Since there is no upper bound on the number of distinct points we can label, this domain will need to be infinite. One such domain appears in Dehlinger et al's verification [21] in the form of lists. Our formalisation is equivalent. We treat a labelling as an assignment from an initial prefix of natural numbers to the points being labelled. Formally:

$$\begin{aligned}\vdash_{def} \texttt{ordering}\, f\, X \iff X = (\{f\, n \mid \texttt{finite}\, X \implies n < |X|\}) \\ \wedge \forall n.\, \forall n'.\, \forall n''.\, (\texttt{finite}\, X \implies n < |X| \wedge n' < |X| \wedge n'' < |X|) \\ \wedge n < n' \wedge n' < n'' \implies \texttt{between}\, (f\, n)\, (f\, n')\, (f\, n'').\end{aligned} \tag{6.4}$$

For generality, we have allowed the set X to be infinite, since the notion of ordering can usefully apply to such sets. Indeed, many of our lemmas about orderings go through in the infinite case, for which the implications in (6.4) with the antecedent $\texttt{finite}\, X$ become vacuous, allowing n to range over the entire set of natural numbers.[1]

With the definition in hand, THEOREM 6 can be formalised in terms of orderings of finite sets.

$$\vdash \texttt{finite}\, X \wedge \texttt{collinear}\, X \implies \exists f.\, \texttt{ordering}\, f\, X.$$

6.3 Natural Numbers

Our definition assumes the existence of natural numbers, but Hilbert was not clear whether he took these to be logically primitive. Veblen, writing some years later, is explicit: "[The axioms] presuppose only the validity of the operations of logic and of counting (ordinal number)" [100, p. 344]. Hilbert *seems* to be making the assumption implicitly when he states the Archimedean Axiom in his Group V, though it could be argued that the following is only meant schematically:

> "If AB and CD are any segments then there exists a *number* n such that n segments CD constructed contiguously from A, along the ray from A through B, will pass beyond the point B." [emphasis added]
>
> [42, p. 26]

[1] In HOL Light, the `finite` sets are defined to be the empty set, and all adjoins to all finite sets.

Euclid does much the same when he expresses the same property using the word "multiplied" (though Euclid mistakes this property for a *definition*):

> "Magnitudes are said to have a ratio to one another which can, when multiplied, exceed one another."
>
> [38, p.114]

Yet elsewhere, Euclid will discuss natural numbers in geometrical terms: his books on number theory literally identify numbers with line segments. Hilbert is faithful to this idea. He shows how to recover arithmetic operations from geometrical figures by exploiting Pascal's and Desargues' Theorems. So why would either want to assume the existence of natural numbers?

The question of foundations here, whether natural numbers are a primitive logical concept required of the theory, or whether they are to be recovered geometrically, is difficult to answer. So when it comes to formalisation, we have tried to base our decisions on the broad philosophical and historical aims of the text. Pasch, Peano, Hilbert and Veblen are all supposed to be rigorising the synthetic geometry of Euclid's *Elements*. There is a story that Euclid, following on from the first crisis in the foundations of mathematics and the discovery of incommensurable magnitudes, would have regarded natural numbers with suspicion, and thus kept them out of his logical foundation [55]. Instead, numbers were to be grounded on secure geometrical notions.

6.3.1 The Axiom of Infinity

The theorem stating that natural numbers exist is derivable in HOL Light assuming a domain *ind* of individuals exists which is Dedekind-infinite, a property formalised as

$$\exists f : ind \to ind.\ \texttt{one\_one}\ f \wedge \neg \texttt{onto}\ f. \tag{6.5}$$

This asserts that there is a one-one but not onto function f on the set of individuals. From it, we can nominate an arbitrary member of the set $\{i : ind \mid \neg(\exists i'.\ f\ i' = i)\}$ to serve as the number 0, nominate f as the successor function, and carve out the natural numbers as the smallest set containing 0 and its own image under f.

The existence of the infinite domain *ind* is not generally assumed as part of classical logic proper, and Russell, in his classic *Principia Mathematica*, took it as a mere antecedent condition on the results which required it [75]. Without the axiom, there are

models of the logic for which all domains — the sets of set theory and the types of simple type theory — each has only finitely many inhabitants.

This should not be the case for the primitive domains of Hilbert's geometry. Hilbert insists as much for his Theorem 7, which states quite plainly that "between any two points on a line there exists an infinite number of points."

Since we need an infinite domain for this theorem anyway, it would be rather elegant if we could derive Axiom 6.5 from Hilbert's order axioms. We can then obtain the natural numbers from the theorem, replacing the abstract type *ind* with a representative type of geometric objects. Our natural numbers will then be quite literally founded in geometry.

Now the axioms of infinity, choice and extensionality are not part of the HOL Light kernel. Instead, they are asserted with `new_axiom`, exactly as we have asserted the axioms of geometry. Thus, to replace the axiom of infinity, we just do not load its theory file. Instead of asserting the axiom, we load the theory files containing our geometric axioms, and *derive it*.

After this, we can reload the usual HOL Light theories which depend on the axiom of infinity, thus reproducing the whole of the HOL Light standard library from a geometric foundation.

6.3.2 Models and a Finite Interpretation

The fact that Hilbert's domains are infinite is not derivable from Group I. We can verify this by exhibiting a finite model of the Group I axioms. To do so, we define a two-place predicate `Group_I` over arbitrary relations l and p. By instantiating the polymorphic types of these relations, one supplies the domains of the interpretation. We then assert a particular model of the axioms, for appropriate l and p, using the formula `Group_I` l p.

Incidentally, we have already defined this predicate and used it to assert the Group I axioms. Having declared our primitive types and our primitive relations, we write

```
new_axiom (Group_I on_line on_plane).
```

We can reuse the predicate `Group_I` to express basic metatheoretical ideas about the Group I axioms, and if we treat the hypothesis as a *context* of axioms, we could even use it to write a *module* of incidence proofs.

In general, there are significant weaknesses to this approach. For one, we are limited in how much we can reason about theories of polymorphic values. As we explained

in Chapter 2, polymorphic types do not increase the expressive power of simple type theory. We cannot quantify over them, and we cannot treat type constructors polymorphically. We would want to do both if we wanted to reason about, say, the theory of monads in Chapter 4. This would require a stronger type theory, such as the extension provided by HOL Omega [44].

For another weakness, when we want to treat `Group_I` as a module of theorems, we would have no convenient way to make new definitions or abstract out new types. This would require a more sophisticated embedding such as the one underlying Isabelle's locales [50].

These issues are not a problem for the specific purposes of this chapter, where we only have to consider a very simple metatheoretical question, but it could cause problems with metatheoretical reasoning in later groups such as Group III where axioms are defined based on quite complex and derived definitions.

6.3.2.1 Verification

A finite model of Group I is realised in the four vertices, six lines, and four planes of a tetrahedron. In this finite interpretation, we can translate all our first-order axioms into propositional theorems and verify them with a tautology checker.

To capture this idea formally, we carved out two finite types. One type is inhabited by four constructors which can be used as interpretations of the four points and the four planes in our model. The other type is inhabited by six constructors, which become our six lines.

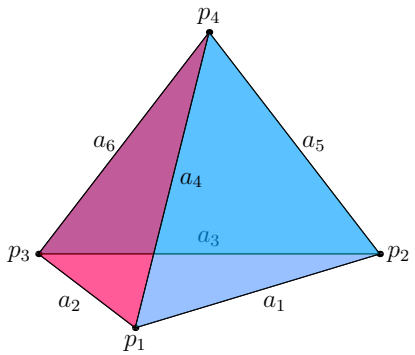
$$\texttt{ps} = \texttt{p1}|\texttt{p2}|\texttt{p3}|\texttt{p4}$$
$$\texttt{lines} = \texttt{l1}|\texttt{l2}|\texttt{l3}|\texttt{l4}|\texttt{l5}|\texttt{l6}.$$

The type definitions are used to automatically derive an abstract type and derive two theorems, an induction theorem and a recursion theorem. For the type `ps`, for instance, we are given

$$\vdash \forall P.\, P\ \texttt{p1} \wedge P\ \texttt{p2} \wedge P\ \texttt{p3} \wedge P\ \texttt{p4} \Longrightarrow \forall p.\, P\ p.$$
$$\vdash \forall p1.\, \forall p2.\, \forall p3.\, \forall p4.\, \exists f.\, f\ \texttt{p1} = p1 \wedge f\ \texttt{p2} = p2 \wedge f\ \texttt{p3} = p3 \wedge f\ \texttt{p4} = p4.$$

The first (induction) theorem can be promoted to an equivalence, and then used to rewrite all universally quantified formulas as finite conjunctions. Similarly, its rewrite

$\vdash_{def}$ `on_line p1 l1` $\wedge$ `on_line p2 l1`
$\wedge$`on_line p1 l2` $\wedge$ `on_line p3 l2`
$\wedge$`on_line p2 l3` $\wedge$ `on_line p3 l3`
$\wedge$`on_line p1 l4` $\wedge$ `on_line p4 l4`
$\wedge$`on_line p2 l5` $\wedge$ `on_line p4 l5`
$\wedge$`on_line p3 l6` $\wedge$ `on_line p4 l6`



$\vdash_{def}$ `on_plane p1 p1` $\wedge$ `on_plane p2 p1` $\wedge$ `on_plane p3 p1`
$\wedge$`on_plane p1 p2` $\wedge$ `on_plane p2 p2` $\wedge$ `on_plane p4 p2`
$\wedge$`on_plane p1 p3` $\wedge$ `on_plane p3 p3` $\wedge$ `on_plane p4 p3`
$\wedge$`on_plane p2 p4` $\wedge$ `on_plane p3 p4` $\wedge$ `on_plane p4 p1`

Figure 6.2: Minimal model

using the infinite DeMorgan rule, $(\forall x.\, P\,x) \iff \neg\exists x.\, \neg(P\,x)$ can be used to rewrite all existentially quantified formulas as finite disjunctions.

Next, by instantiating the universally quantified variables in the second (recursion) theorem with the first four natural numbers respectively, we can prove that `p1`, `p2`, `p3` and `p4` are mutually distinct. From this, every valid first-order formula over the types `ps` and `lines` can be rewritten to a propositional formula and then quickly verified.

To verify our model, we inductively define the incidence predicates `on_line` and `on_plane` over the types `ps` and `lines`, as shown in Figure 6.2. We then verify the following theorem in HOL Light

$$\vdash \texttt{Group1 on\_line on\_plane}.$$

After unfolding the definitions of `on_line` and `on_plane`, it turns out that the theorem can be proven by equational reasoning alone. We replace the universals and existentials with conjunctions and disjunctions respectively. After simplifying, the remaining goals require us to show that points, lines or planes are distinct, which is dealt with by the recursion theorem.

The same method allows us to formally deal with the case of a weakened Axiom I, 3.1 that we mentioned briefly in §3.2.2. We just define a seven element finite set for the lines, and use the same inductive definitions for `on_line` and `on_plane`, leaving the

seventh line "dangling" without any incident points. Again, by rewriting the axioms propositionally and simplifying, we can show that this is a (presumably inappropriate) model for the weakened axioms, and thus justify our formalisation of Axiom I, 3.1.

To show that the tetrahedral model is minimal, we verified that there exist at least the four points, six lines and four planes satisfying the conditions we gave to inductively define `on_line` and `on_plane` in the model. To do this, we use Theorem 3.1 which gives us three distinct points on a plane. We then use Axiom I, 8, showing that there is a fourth point not on this plane. The remaining axioms are then sufficient to connect each pair of points by a unique line, and every triple of points by a unique plane.

6.4 Infinity

Once in Group II, there are only infinite models. Indeed, we can just apply Axiom II, 2 repeatedly to obtain an arbitrary number of points. That is, starting with distinct points A and B, we can obtain points A, B, C, D, E, ..., Y, Z, satisfying

$$\texttt{between}\ A\ B\ C$$
$$\texttt{between}\ A\ C\ D$$
$$\texttt{between}\ A\ D\ E$$
$$\ldots$$
$$\texttt{between}\ A\ Y\ Z.$$

If we apply THEOREM 5, then we can move from theorems such as `between` $A\ B\ C$ and `between` $A\ C\ D$ to `between` $A\ B\ D$, and so can prove that the points above are mutually distinct. Therefore, for any number n, we can prove a theorem saying there must be n distinct points. The domain of interpretation must be infinite.

6.5 A Geometric Successor

The function which is said to exist in the axiom of infinity is a successor function. When we derive it as a theorem, we shall obtain its witness, and, staying faithful to geometry, we shall use a witness that is effectively a function from a point to a distinct point on a line segment. The witness is based on the diagram in Figure 6.3.

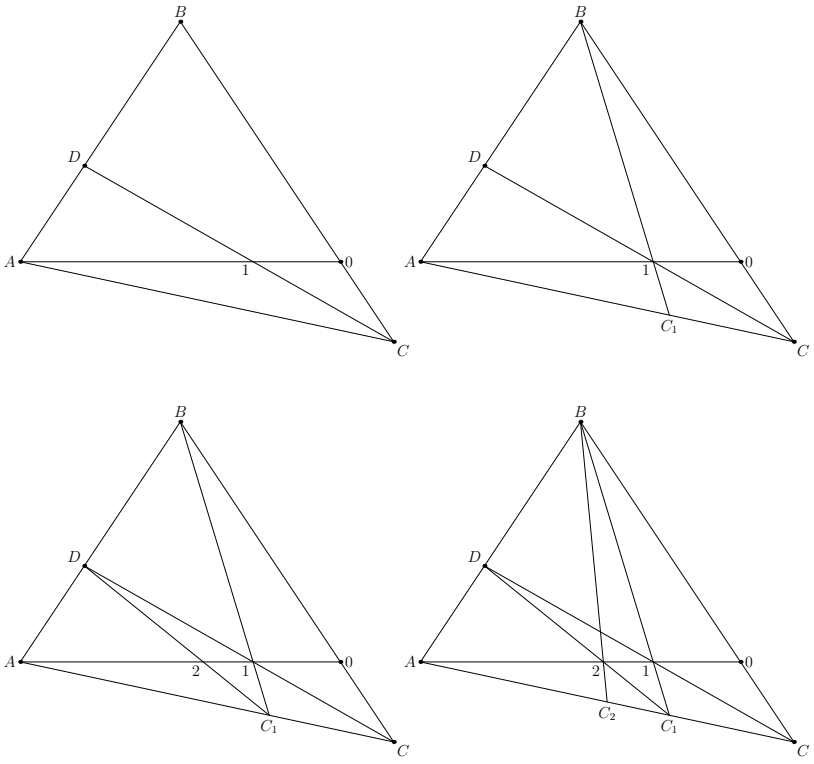


Figure 6.3: The successor function

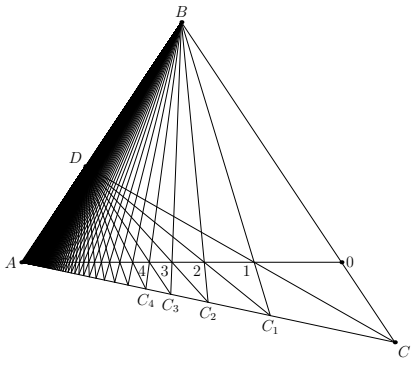


Figure 6.4: Successors tending to A

Formally, the diagrams are the sets of points satisfying the following formalised constraint

$$\begin{aligned}\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ 0\ a)\\ \wedge\, \texttt{between}\ A\ D\ B \wedge \texttt{between}\ B\ 0\ C\\ \wedge\, (\texttt{between}\ A\ N\ 0 \vee N = 0).\end{aligned} \tag{6.6}$$

Our natural numbers are carved out from the set of all diagrams satisfying this constraint. We abstract this set into a type (`ind`), giving us two functions: an abstraction function `mk_ind` which promotes any diagram into the abstract type, and a representation function `dest_ind` which converts an inhabitant of the abstract type into its diagram representation. In order for this to be sound, we only need to prove that the type will have at least one inhabitant. This is easily settled, since the diagram for 0 is constructed in the proof of THEOREM 3 (§5.1).

Each diagram is represented by six points, five of which are fixed by the successor function, while the sixth point N starts at 0 and moves step-by-step towards A. The points C_1, C_2 and C_3, shown in Figure 6.3, are not explicitly represented in our type but can be determined as the intersection of BN and CD.

Informally, and for the purposes of explaining our formalisation, we identify a diagram up to a relabelling of the first five points, and thus identify it with its sixth point. In this way, when we talk of the object 0, we may be referring to the *diagram* 0, which is the six-tuple representing inhabitants of `ind`, or to the *point* 0, which is the sixth component of this six-tuple. Similarly, we shall talk about the *diagram* that is the successor of 0, as well as the *point* that is the successor of 0. This is just for convenience here. The formalisation itself is always unambiguous.

Thus, the successor of 0 is the diagram obtained by replacing 0 with 1, the intersection of CD and $A0$. To obtain the next successor, we first find the intersection of $1B$ and AC, namely the point C_1. We then replace 1 with the intersection of C_1D and $A0$.

In general, the successor of a diagram is obtained by finding C', the intersection of BN and AC, and then finding the intersection of $C'D$ and $A0$. The definition is formalised in Figure 6.5.

We have used the ι operator in this definition. This is the "definite description" operator, which is a weaker version of the ε "indefinite description operator". The ε operator is specified by one of the three classical axioms of higher-order logic, and is equivalent to the full axiom of choice.

$$\vdash \forall P.\ \forall x.\ P\ x \implies P(\varepsilon x.\ Px).$$

$$\begin{aligned}\vdash_{def} \texttt{ind\_suc}\, n = {} & \texttt{let}\ (A,B,C,D,0,N) = \texttt{dest\_ind}\, n\ \texttt{in} \\ & \texttt{mk\_ind}(A,B,C,D,0,\iota S. \\ & \quad \exists C'.\ (\exists a.\ \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C'\ a \wedge \texttt{on\_line}\ N\ a) \\ & \quad \wedge (\exists a.\ \texttt{on\_line}\ C'\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ S\ a) \\ & \quad \wedge (\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ C'\ a) \\ & \quad \wedge \texttt{between}\ A\ S\ 0 \wedge \texttt{between}\ A\ S\ N).\end{aligned}$$

Figure 6.5: Definition of `ind_suc`

From ε, we can define the ι operator, which requires that the predicate P is satisfied by exactly one value.

$$\vdash (\iota x.\ P\ x) = \varepsilon x.\ P\ x \wedge \forall y.\ P\ y \implies x = y.$$

By using this operator and avoiding the somewhat controversial stronger axiom of choice, we feel we are in a better position to argue that we have recovered the axiom of infinity in a logically "secure" way. The images of our successor function are uniquely defined from their predecessors, and the natural numbers themselves can be uniquely carved out of the type `ind`. There is exactly one object 0 up to relabellings of the points in our figures, not an arbitrary set of possibilities in an abstract type from which we choose one example.

The price comes in the complexity of the `ind` representatives. We cannot simply define an infinite domain of points using Axiom II, 2 as we did at the beginning of this section. We need enough information in each of our figures to constrain the possible placement of successors relative to their predecessors.

6.5.1 Lemmas

Our successor function above destructs an abstract diagram into its six points, chooses the unique point S, and finally rebuilds the diagram.

We now need some lemmas concerning `dest_ind`. Importantly, we need to show that the reconstructed diagram represents an abstract diagram, and thus show that the image of `mk_ind` in our definition is well-defined. The formal theorem is given in Figure 6.6.

$$\begin{aligned}\vdash \texttt{dest\_ind}\,(\texttt{ind\_suc}\,n) = \texttt{let}\ \ (A,B,C,D,0,N) = \texttt{dest\_ind}\,n\ \ \texttt{in}\\ (A,B,C,D,0,\iota S.\\ \exists C'.\,(\exists a.\,\texttt{on\_line}\,B\,a \wedge \texttt{on\_line}\,C'\,a \wedge \texttt{on\_line}\,N\,a)\\ \wedge\,(\exists a.\,\texttt{on\_line}\,C'\,a \wedge \texttt{on\_line}\,D\,a \wedge \texttt{on\_line}\,S\,a)\\ \wedge\,(\exists a.\,\texttt{on\_line}\,A\,a \wedge \texttt{on\_line}\,C'\,a \wedge \texttt{on\_line}\,D\,a)\\ \wedge\,\texttt{between}\,A\,S\,0 \wedge \texttt{between}\,A\,S\,N).\end{aligned}$$

Figure 6.6: Images of `ind_suc` are are well-defined

The key step needed to verify this theorem comes from realising that the diagrams involving the points $A,B,C,0,D$ and A,B,C',N,D satisfy the same constraints. So in each case, we can apply Pasch's axiom (II, 4) to $\triangle AB0$ and the line $C'D$ to obtain a point S between A and 0. For our initial diagram, where $C' = C$, we can apply this argument directly. For the other diagrams, we just need to find the point C'.

To do this, we apply Pasch's axiom (II, 4) to $\triangle A0C$ and the line BN, to place the point C' between A and C. We can then locate S between A and N. Finally, since N is between A and 0, THEOREM 5 shows that the point S must also lie between A and 0.

6.6 Theorem of Infinity

Finally, we must verify that our function is one-one but not onto. Verifying that `ind_suc` is one-one just means verifying that `ind_suc` $m =$ `ind_suc` n always implies that $m = n$. Geometrically, this means identifying points in a diagram, and with our discoverers this turned out to be easy. Using the tactic `discover_tac by_eqs`, the incidence automation could find all the necessary equalities that arise from the assumption `ind_suc` $m =$ `ind_suc` n automatically.

We verified the fact that `ind_suc` is not onto declaratively. The basic verification works by noting that 0 defines the first diagram, while all images of `ind_suc` use a point S which is defined to be strictly between A and 0. Thus, 0 is not in the image of the function.

Putting the two facts together, we verify:

$$\vdash \texttt{one\_one ind\_suc} \wedge \neg \texttt{onto ind\_suc}.$$

This tells us that the abstract type `ind` has an infinite domain. However, the domain is *not* isomorphic to the natural numbers. It consists at least of the diagrams where the point N is any point lying between A and 0, and infinitely many of these are not in the sequence

$$0, \texttt{ind\_suc}\ 0, \texttt{ind\_suc}\ (\texttt{ind\_suc}\ 0), \texttt{ind\_suc}\ (\texttt{ind\_suc}\ (\texttt{ind\_suc}\ 0)), \ldots.$$

For instance, no point between 0 and `ind_suc` 0 is in this sequence.

To remove the unwanted diagrams, we follow HOL Light's construction of the natural numbers, which inductively restricts us to the smallest closure of the successor function starting from 0.

6.7 THEOREM 6 Revisited

With the natural numbers defined, our definition of ordering (6.4) goes through. We can now verify THEOREM 6. We do it in two parts.

6.7.1 At Least One Ordering

Hilbert notes that THEOREM 6 is a generalisation of THEOREM 5, and it turns out that we prove it by generalising the last part of the proof of THEOREM 5. We did not give this part of the proof in the previous chapter. We give it now in Figure 6.7

For the purposes of verification, we probably want to tidy up Hilbert's case-analysis. We first take Hilbert's last three clauses as a nested case-analysis. Then the last case is contradictory and can be discarded. We are left with:

$$\begin{cases} \text{R lies between P and S,} \\ \text{P lies between R and S,} \\ \text{S lies between P and R,} \begin{cases} \text{Q lies between P and S,} \\ \text{S lies between P and Q.} \end{cases} \end{cases}$$

Hilbert says the case-analysis arises from applications of THEOREM 4, which tells us that of three points, one lies between the other two. When we generalise from four points to $n+1$ points, we apply induction, and two of the applications of THEOREM 4 become applications of our inductive hypothesis to n points. We give the generalised

Now let any four points on a line be given. Take three of the points and label Q the one which by THEOREM 4 and Axiom II, 3 lies between the other two and label the other two P and R. Finally, label S the last of the four points. By Axiom II, 3 and THEOREM 4 again it follows then that the following five distinct possibilities for the position of S exist:

R lies between P and S,
or P lies between R and S,
or S lies between P and R simultaneously when Q lies between P and S,
or S lies between P and Q,
or P lies between Q and S.

The first four possibilities satisfy the hypotheses of [the second lemma] and the last one satisfies those of [the first lemma]. THEOREM 5 is thus proved.

[42, p. 7]

Figure 6.7: Case-analysis for THEOREM 5

proof now together with some verified lemmas. We follow the original proof closely to show how the inferences become generalised.

Proposition (THEOREM 6)**.** *Given any finite number of points on a line it is always possible to label them A, B, C, D,...,K in such a way that the point labelled B lies between A, and C, D, E,...,K, the point labelled C lies between A, B and D, E,...K, D lies between A, B, C and E,..., etc.*

Proof. The case for 1 and 2 points is trivial. The case for three points is covered by THEOREM 4. Assume, for induction, that the theorem holds for n points.

Now let $n+1$ points on a line be given. Take n of the points and label P, Q_1, Q_2, ..., Q_{n-2}, R the ones which by our inductive hypothesis are ordered along the line. Finally, label S the last of the $n+1$ points. By Axiom II, 3 and THEOREM 4 it follows then that the following three distinct possibilities for the position of S exist:

R lies between P and S,
or P lies between R and S,
or S lies between P and R.

In the last case, we apply our inductive hypothesis to the points P, Q_1, Q_2, …, Q_{n-2}, S. For any of the positions of S, we can apply the following theorem (6.7) to label all the points in order:

$$\begin{aligned} &\vdash \texttt{finite}\, X \wedge \texttt{ordering}\, f\, X \wedge \texttt{between}\, (f0)\, (f(|X|-1))\, x \\ &\qquad \Longrightarrow \exists g.\, \texttt{ordering}\, g\, (\{x\} \cup X) \end{aligned} \tag{6.7}$$

$$\vdash_{def} \texttt{bounds}\, P\, Q\, X \iff P, Q \in X \wedge \forall R.\, R \in X - \{P, Q\} \Longrightarrow \texttt{between}\, P\, R\, Q$$

$$\begin{aligned} &\vdash x \in X \wedge \texttt{finite}\, X \wedge \texttt{ordering}\, f\, X \wedge \texttt{bounds}\, P\, Q\, X \\ &\qquad \Longrightarrow \exists f'.\, \texttt{ordering}\, f'\, X \wedge f'\, 0 = P \wedge f'\, (|X|-1) = Q. \end{aligned} \tag{6.8}$$

□

We have replaced Hilbert's reference to his earlier parts of THEOREM 5 with a reference to its generalisation in Theorem 6.7. We actually need a few other theorems. Consider that after our final application of the inductive hypothesis, we will have two orderings f and g. The ordering f runs P, Q_1, Q_2, …, Q_{n-2}, R, and the ordering g runs P, Q_1, Q_2, …, Q_{n-2}, S. But this is a notational shortcut, as we gloss over the implicit assumption that we have chosen the orders such that $f(0) = g(0) = P$. In our verification, we can make this reasoning explicit with Theorem 6.8, which uses the auxiliary concept of `bounds`.

Our generalised proof handles its case-analyses by one application of THEOREM 4, and two applications of the inductive hypothesis. This means we must apply well-founded induction rather than normal structural induction, since we must instantiate our inductive hypothesis in two different ways.

The base case of the induction is captured in two lemmas:

$$\vdash \exists f.\, \texttt{ordering}\, f\, \emptyset.$$

$$\vdash \texttt{collinear}\{x, y, z\} \Longrightarrow \exists f.\, \texttt{ordering}\, f\, \{x, y, z\}.$$

The case for the empty set is trivial: any function witnesses the existential, since the conditions on the function are all vacuous. The second case actually breaks down into four separate cases, depending on whether any of the x, y or z are equal. For a single point, we can pick the constant function to that point, and for two points $x \neq y$, we pick the function which maps 0 to x and everything else to y.

For three points `between` P Q R, we have the ordering which maps 0 to P, 1 to Q and all other numbers to R. Since THEOREM 4 requires that, of any three collinear points,

one lies between the other two, it follows that there must be an ordering for any three collinear points. This takes care of the base case. And thus, we formally verify:

$$\vdash \texttt{finite}\, X \wedge \texttt{collinear}\, X \implies \exists f.\, \texttt{ordering}\, f\, X. \qquad \text{(THEOREM 6)}$$

6.7.2 Exactly Two Orderings

Hilbert closes his statement of THEOREM 6 with a remark that had a challenging verification: "Besides this order of labelling there is only the reverse one that has the same property.".

To verify this, we began with a lemma: if we have two orders f and g where $f(0) = g(0)$, then the orders are identical.

$$\vdash \texttt{ordering}\, f\, X \wedge \texttt{ordering}\, g\, X \\ \wedge f\, 0 = g\, 0 \wedge (\texttt{finite}\, X \implies n < |X|) \implies f\, n = g\, n.$$

The verification of this theorem uses induction. Aiming for a contradiction, we assume that $f\ (n+1) \neq g\ (n+1)$ and then consider the relative positions of $f\ 0$, $f\ (n+1)$ and $g\ (n+1)$ that arise from THEOREM 4. For each possibility, we can apply THEOREM 5 to show that there will end up being a point in the ordering between $f\ n$ and $f\ (n+1)$, or a point between $g\ n$ and $g\ (n+1)$. Both are impossibilities.

Putting these facts together, we can verify Hilbert's assertion.

$$\vdash \texttt{finite}\, X \wedge \texttt{ordering}\, f\, X \wedge \texttt{ordering}\, g\, X \wedge (\texttt{finite}\, X \implies n < |X|) \\ \implies \forall n.\, f\, n = g\, n \vee f\, n = g(|X| - n - 1).$$

We mention this only briefly, because we never apply this theorem in the later verification.

6.8 An Ordering Tactic

THEOREM 6 says everything one wants to know about the order of a finite number of points on a line, but it is not immediately obvious how to apply it.

One thing we can do with THEOREM 6 is use it to convert problems involving betweenness into problems of natural numbers. To handle this, we will need to consider

$$\vdash \texttt{finite}\ X \wedge \texttt{collinear}\ X$$
$$\implies \exists f.\ \forall A.\ \forall B.\ \forall C.\ A \in X \wedge B \in X \wedge C \in X$$
$$\implies \left(\begin{array}{l} \texttt{between}\ A\ B\ C \\ \quad \iff (f\ A < f\ B \wedge f\ B < f\ C) \vee (f\ C < f\ B \wedge f\ B < f\ A) \end{array} \right)$$
$$\wedge \forall A.\ \forall B.\ A \in X \wedge B \in X \implies (A = B \iff f\ A = f\ B).$$

Figure 6.8: Rewriting betweenness to inequalities.

what is basically the inverse of the `ordering` function, given as f in the theorem in Figure 6.8.

On the assumption that one has a collinear and finite set of points X, this theorem allows us to obtain a function f with which we can take goals in terms of betweenness and equalities of points, and rewrite them into inequalities and equalities of natural numbers. Once rewritten, the goal can be solved by HOL Light's decision procedure for linear arithmetic. The procedure is not particularly efficient, but in practice, we only consider simple betweenness problems (at most, ones involving six points).

For convenience, we have implemented a procedural tactic `ORDER_TAC`, which is parameterised on a finite set enumeration (a term of the form $\{P_1, P_2, \ldots, P_n\}$). The tactic instantiates the variable X in the above theorem with the enumeration, uses rewriting to prove that X is finite, and then uses the incidence discoverer from Chapter 4 to prove that it is collinear. It then obtains the function f and uses it to rewrite the goal. Finally, it hands over to HOL Light's decision procedure for linear arithmetic, `ARITH_TAC`.

6.8.1 Example

To round up this chapter, we will demonstrate the use of our linear reasoning tactic `ORDER_TAC` by applying it to another theorem which says that there is an infinite number of points between any two others points. We have already effectively proven this theorem using the Dedekind definition of infinite, but here, we use the HOL Light predicate `infinite` which is just the complement of the recursively defined predicate `finite`.

> "THEOREM 7. Between any two points on a line there exists an infinite number of points."

[42, p. 8]

$$\vdash P \neq Q \implies \texttt{infinite}\,\{R \mid \texttt{between}\,P\,R\,Q\}.$$

We start our proof by assuming that the set of points between P and Q is finite. We then consider separately whether the set contains fewer than two elements, or whether it contains more than two elements.

We describe the proof briefly. In the first case, we just use THEOREM 3 twice to find two points between P and Q, from which we can obtain a contradiction. In the second case, we get to apply THEOREM 6. The basic idea is as follows: we obtain an ordering f of all points between P and Q, and then take the first two elements of this ordering, namely $f\,0$ and $f\,1$. Via THEOREM 3, we can find a point R that lies strictly between them. According to our assumption, this point must be in the image of f. But this contradicts the definition of an ordering.

The part of our verification where we apply our linear ordering tactic might come as a surprise. It is actually used to verify a point glossed over in the last paragraph, namely that R must be in the image of f. To show this, we must verify that R lies between P and Q.

Before we had implemented our tactic, we tried to verify this matter directly using THEOREM 4 and THEOREM 5, but we gave up. The necessary case-analyses were just not intuitive to us. But `ORDER_TAC` takes care of the matter elegantly.

$$\begin{array}{ll}
\texttt{so consider}\,R\,\texttt{such that between}\,(f\,0)\,R\,(f1) & 7\\
\texttt{have between}\,P\,(f\,0)\,Q \wedge \texttt{between}\,P\,(f\,1)\,Q\,\texttt{from}\,6\,\texttt{by}\ldots & 8\\
\texttt{hence between}\,P\,R\,Q\,\texttt{from}\,6,7\,\texttt{using ORDER_TAC}\,\{P,Q,R,f\,0,f\,1\} &
\end{array}$$

6.9 Conclusion

This chapter has been concerned with the theory of linear-ordering based on Hilbert's three-place `between` relation, culminating in a tactic for solving problems arranging points along a line by reducing them to a decision procedure for linear problems in natural numbers. The use of natural numbers was explicitly permitted in Veblen's ordered geometry [100], but not in Hilbert's. We have closed the gap by showing

how they can be recovered in higher-order logic from geometry without the axiom of infinity.

We provided another way to deal with ordering problems, by interpreting Hilbert's THEOREM 6 as a metatheorem, and finding a way to express linear problems as efficient stacks of betweenness formulas. The approach is more limited, since it assumes that all points considered in the problem are distinct, and we found that this assumption is too strong in practice, but we might expect it to have better performance in some cases, since it is more carefully tailored to Hilbert's geometry. We leave analysis of this matter to future work, and use the reduction to linear arithmetic for the rest of our verification.

We have now covered all the automation we will need for our verification of the Polygonal Jordan Curve Theorem: in summary, we use a search algebra to handle the implicit incidence reasoning from Group I, and a tactic to handle linear reasoning from Group II. These two automated tools will be used extensively for the verifications we discuss in the remaining chapters.

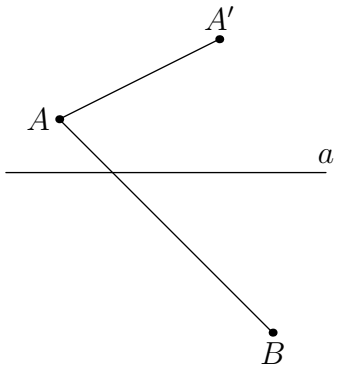
Chapter 7

Ordering in the Plane

In the last chapter, we used linear arithmetic to settle problems of points ordered along a line. But what if we want to reason about the relative positions of points in a *plane*? For this, we note that a line partitions a plane into two sides. So we can compare the relative position of points in the plane by asking on which side they are on. These *sides* are defined in the *Grundlagen der Geometrie* in Hilbert's next result, THEOREM 8, which he cites as the key theorem needed for the Polygonal Jordan Curve Theorem.

7.1 Definitions and Formalisation

THEOREM 8. Every line a that lies in a plane α separates the points which are not on the plane α into two regions with the following property: Every point A of one region determines with every point B of the other region a segment AB on which there lies a point of the line a. However any two points A and A' of one and the same region determine a segment AA' that contains no point of a.



DEFINITION. *The points A, A' are said to lie in the plane α on one and the same side of the line a and the points A, B are said to lie in the plane α on different sides of a.*

> DEFINITION. Let A, A', O, B be four points of the line a such that O lies between A and B but not between A and A'. The points A, A' are then said to lie *on the line a on one and the same side of the point O and the points A, B are said to lie on the line on different sides of the point O. The totality of points of the line A that lie on one and the same side of O is called a* ray *emanating from O. Thus every point of a line partitions it into two rays.*
>
> [42, p. 8]

Though we shall not need the notion of rays in our verifications, we shall describe some of its formalisation. The definition is effectively the one-dimensional analogue of THEOREM 8, and we will use it to explain the general approach to verifying that theorem. We also mention that *rays* make for a useful abstraction in later definitions and axioms (see §7.1.1 below).

Now for rays, a three-place relation `same_side` relating a point with every other pair of points on a line is redundant when we can just relate a ray with its incident points. So we drop the relation. Similarly, we drop the relation for planes and instead introduce the two-dimensional analogue of rays, namely *half-planes*. A half-plane will be the totality of points on the same side of a line a in the plane α. So when we say that two points lie on the same side of the line a in the plane α, we shall mean that they lie on a single half-plane in α and bounded by a.

7.1.1 Rays

We briefly justify keeping the definition of rays in our formalisation. Rays are useful in Group III, where Hilbert introduces an axiom governing congruence of angles:

> Let α be a plane and h, k any two distinct rays emanating from O in α and lying on **distinct lines**. The pair of rays h, k is called an *angle* and is denoted by $\angle(h,k)$ or by $\angle(k,h)$.
>
> ...
>
> III, 4. *Let $\angle(h,k)$ be an angle in a plane α and a' a line in a plane α' and let a definite side of a' in α' be given. Let h' be a ray on the line a' that emanates from the point O'. Then there exists in the plane α' one and only one ray k' such that the angle $\angle(h,k)$ is congruent or equal to the angle $\angle(h',k')$ and at the same time all interior points of the angle $\angle(h',k')$ lie on the given side of a'.*
>
> [42, p. 11]

Notice how much more complicated Hilbert's axioms have become by this group. Here, we have an axiom which juggles eight geometric entities, six of which are not even primitive. It is easy to make a mistake here and we recommend that, if this axiom is to be reliably formalised, that the notions of *angle*, and thus, the dependent notion of *ray* must be fully formalised and a decent theory developed before we can trust that the definitions and axiom are correct. In simple type theory, one can gain further confidence by introducing rays as an abstract type, which can simplify the formalisation of Axiom III, 4 by pushing constraints into the type-checker (see earlier work [86]).

7.1.2 Quotienting

With a slight clarification, the "same side" relations in Hilbert's definitions define equivalence relations, and rays and half-planes emerge as the equivalence classes. In particular, the relation "same side of the point O" quotients the set of points in space other than a point O into the set of all rays emanating from O, or alternatively, with *origin* O. Similarly, the relation "same side of the line a" quotients the set of points in space not on the line a into the set of all half-planes bounded by a. Note that we do not restrict the dimension here, and thus allow rays and half-planes to emanate in all directions in space.

We have had to fill in an ambiguity in Hilbert's definition, and exclude the closure points or boundary from both rays and half-planes. If we include the point O for a ray, or the line a for a half-plane, and we allow an arbitrary point to be on the same side of O or a, then we will have only one equivalence class: the whole of space. If we include O and a in every ray and half-plane, but declare all other points to be on a different side of O and a, then our equivalence classes tell us that the set $\{O\}$ counts as a zero-dimensional ray, while the line a counts as a one-dimensional half-plane. We exclude these possibilities, and thus make all rays and half-planes, as equivalence classes, open sets: a ray does not include its origin and a half-plane does not include its boundary. Incidentally, Poincaré made the same decision, remarking parenthetically in his review of Hilbert "I add, for precision, that I consider [the origin] as not belonging to either [half-ray]" [39, p. 11].

7.1.3 Automatic Lifting

HOL Light has several powerful procedures for automatically dealing with quotienting and producing a strong type for the quotient sets. Assuming that $(\equiv) : \tau \to \tau \to \tau$ is an equivalence relation on τ, there is a procedure which splits τ into equivalence classes. A new abstract type is then introduced in the theory, isomorphic with the class of all these equivalence classes. Additional procedures then exist which allow the user to *lift* HOL functions which are provably well-defined for the equivalence relation to the abstract type. We wanted to use these facilities to introduce the new abstract types of rays and half-planes, and so introduce our primitive relations on these abstract types by lifting well-defined relations.

Unfortunately, we do not have types for the domains of the equivalence relations. The "same side" relations Hilbert defines are only equivalence relations on families of subsets of space. Our equivalence relations for rays are indexed by a point O and have as domain the set of points in space minus O. Our equivalence relations for half-planes are indexed by a line a and have as domain the set of points in space minus a. Simple type theory does not allow us to consider these families at the type-level.

We were not sure how best to tackle this problem, and we have not implemented a generic solution. Instead, in this section, we review one possible strategy which takes us to our new quotiented type via an intermediate type. Here, we will only consider the strategy applied to rays. The half-planes case is exactly analogous.

7.1.3.1 Intermediate Types

For any point O, we must consider the set of all points P in space which are not on O. We can do this by creating an abstract type represented by pairs (O,P) where both components are distinct. We call this type `arrow`. It is the type of directed line-segments, or *arrows* $\overrightarrow{OP}$, where $O \neq P$. The origin of the arrow is the point O, and the arrow *points* in the direction P. By defining this type, we obtain abstraction and representation functions:

$$\texttt{mk\_arrow} : (\texttt{point},\texttt{point}) \to \texttt{arrow}.$$
$$\texttt{dest\_arrow} : \texttt{arrow} \to (\texttt{point},\texttt{point}).$$

These are similar to `mk_ind` and `dest_ind` from the last chapter. They map back-and-forth between pairs of points and the arrows they represent.

The relation "same side of" can now be reinterpreted on these arrows. Our relation will effectively ask whether two arrows have the same position and direction. With some abuse of HOL Light notation (we pretend that we can extract the two endpoints of an arrow with a *pattern match*), we have:

$$\begin{aligned} &\texttt{equiv\_arrow} : \texttt{arrow} \to \texttt{arrow} \to \texttt{bool} \\ \vdash_{def} \texttt{equiv\_arrow}\ &\overrightarrow{OP}\ \overrightarrow{O'Q} \\ &\iff O = O' \wedge (P = Q \vee \texttt{between}\ O\ P\ Q \vee \texttt{between}\ O\ Q\ P). \end{aligned} \tag{7.1}$$

We now just verify that this relation is an equivalence relation on the type of arrows:

$$\begin{aligned} &\vdash \texttt{equiv\_arrow}\ s\ s \\ &\quad \wedge (\texttt{equiv\_arrow}\ s\ t \iff \texttt{equiv\_arrow}\ t\ s) \\ &\quad \wedge (\texttt{equiv\_arrow}\ s\ t \wedge \texttt{equiv\_arrow}\ t\ u \implies \texttt{equiv\_arrow}\ s\ u). \end{aligned}$$

The only challenge when verifying this theorem is dealing with transitivity. In our earlier work [86], where we tried to define rays as equivalence classes without using any automatic quotienting, the verification took some hard pen-and-paper work before we could transcribe it. We were bogged down with picky variable instantiations needed to apply THEOREM 5, made worse by the disjunction in our definition (7.1) which throws up several case-splits. But in our HOL Light development, we have the linear reasoning tactic from the last chapter, which makes the matter trivial. It automatically deals with the case-splits, and can solve the goal without any explicit reference to other theorems.

7.1.3.2 Lifting to a Theory of Rays

With the equivalence relation verified, it is a simple matter to define the quotient type of rays. With the command

```
define_quotient_type "ray" ("mk_ray","dest_ray") equiv_arrow
```

we introduce a new type `ray` into the theory, together with their abstraction and representation functions `mk_ray` and `dest_ray`.

The great thing about HOL Light's quotienting facilities is that we do not need to deal with these particular abstraction and representation functions directly. When we lift theorems the functions are used automatically.

However, HOL Light will not automatically plumb theorems about the endpoints of arrows through our intermediate type and lift them to our type of rays. Consider the relation which says that a given point lies on a given ray. If rays were an equivalence class on the space of all points, this relation would be lifted directly from the partially applied equivalence relation. Here, we must instead build an arrow, using the abstraction function for arrows, namely `mk_arrow`.

$$\begin{aligned}\vdash_{def} \texttt{on\_ray\_of\_arrow}\ P\ \overrightarrow{OQ}\\ \iff P \neq O\\ \wedge\, \texttt{equiv\_arrow}\ \overrightarrow{OQ}\ (\texttt{mk\_arrow}\ (O,P)).\end{aligned}$$

It is trivial to verify that this relation is well-defined, but to use it effectively in proofs relating points of an arrow to points on the ray of an arrow, we need to manually fold and unfold the definition of arrows. It is tedious enough to verify that

$$\vdash \texttt{on\_ray\_of\_arrow}\ P\ \overrightarrow{OQ} \iff P = Q \vee \texttt{between}\ O\ P\ Q \vee \texttt{between}\ O\ Q\ P.$$

We suspect that the creation of the intermediate `arrow` type and the lifting of theorems through this type could be automated, greatly improving the presentation of the theory. This would amount to an extension of HOL Light's quotienting facilities, which would allow it to handle indexed families of equivalence relations such as that of arrows lying in the same direction, indexed by a point of origin. This would make for suitable future work, and we believe that Hilbert's geometry offers a nice example of where such facilities would be useful.

7.2 Theory of Half-Planes

The theory of rays is largely trivial when we have our linear reasoning tactic. Everything we want to know about linear order is bound up in THEOREM 6, from which that tactic was derived. Two-dimensional order is less straightforward. We get this impression from Hilbert himself, who justifies the definition of half-planes with a distinguished theorem (THEOREM 8). He gives no corresponding theorem for rays, but instead, just assumes his definition is sound.

As with the theory of rays, our theory is based on lifting from an intermediate type. We mediate the notion of half-plane by a line and a point not on that line, where a ray was mediated by a point and a distinct point. The half-plane intermediary lacks

the pleasing geometric interpretation of *arrows*, but the basic plumbing and proofs are similar.

$$\begin{aligned} \vdash_{def} \texttt{equiv\_half\_plane}\ (P,a)\ (Q,b) \\ \iff\ & a = b \\ & \wedge (\exists \alpha.\ \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha \\ & \qquad \wedge \forall S.\ \texttt{on\_line}\ S\ a \implies \texttt{on\_plane}\ S\ \alpha) \\ & \wedge \neg \texttt{on\_line}\ P\ a \wedge \neg \texttt{on\_line}\ Q\ a \\ & \wedge \neg (\exists R.\ \texttt{on\_line}\ R\ a \wedge \texttt{between}\ P\ R\ Q). \end{aligned} \tag{7.2}$$

The equivalence relation is unfortunately convoluted by constraints, since the main property saying when two points are on the same side of a line is a negative one and thus quite weak. The correctness may not be immediately obvious. If the reader needs more confidence than simple inspection provides, they can perhaps be assured by the fact that we have derived many of the expected theorems about half-planes.

Many of the theorems are trivial, and are only provided to link the primitive types `point`, `line` and `plane` with our new type of `half_plane`. The two non-trivial theorems we need to verify are, firstly, that the relation defined above is transitive, and secondly, that there are exactly two half-planes to each plane. As we shall see, both theorems together can be understood as a strengthening of Pasch's Axiom (II, 4).

7.2.1 Transitivity

Consider the transitivity problem. Suppose that the points A and B are on the same side of the line a, and that B and C are also on the same side. We must show that A and C are then on the same side.

According to our definition (7.2), this means we must show that if the line a does not intersect between A and B, and does not intersect between B and C, then it cannot intersect between A and C. Equivalently, if there is an intersection at A and C, then there is an intersection either between A and B or between B and C. This is already very close to Pasch's axiom.

Pasch's axiom (II, 4) asserts that, given a triangle ABC, if a line a lies in the plane of ABC and crosses the side of a triangle, and does not intersect a vertex, then it must leave by one of the other two sides.

We have most of these assumptions in place. We know that our points A, B and C are planar: that assumption was made part of the definition (7.2). We know that the line a

does not meet any vertex, since this is a defining requirement of any representative of our intermediate type. The only assumption we have not met is that A, B and C form a triangle.

But actually, this assumption on Pasch's axiom is not necessary. The conclusion holds even if A, B and C lie on a line, though we have not been able to prove it up until now. The verification, which uses THEOREM 6 via our linear reasoning tactic, is given in Figure 7.1.

```
assume on_line P a ∧ ¬on_line C a ∧ between A P B                    0
assume ∃a. on_line A a ∧ on_line B a ∧ on_line C a                   1
take P
thus on_line P a from 0
have C ≠ P from from 0
hence between A P C ∨ between B P C using ORDER_TAC {A,B,C,P} from 0,1
```

Figure 7.1: Pasch's Axiom when A, B and C are collinear

In the proof in Figure 7.1, we have pared down the assumptions significantly. Now that we assume that the three points are collinear, there is no need to mention planes. Without the planes, the only remaining assumption is the one which says that the line a does not meet any vertex. In verifying the theorem, we initially thought to throw out this assumption, believing it was as unnecessary as the planar assumption, but our linear reasoning tactic thought otherwise. It promptly told us that the resulting situation entails no contradiction. It will not give us a valid model, but with a moment's thought, we realise that if $C = P$, then the strictness of the `between` relation means that the conclusion cannot possibly hold.

To fix this, we add back the assumption that C is not on the line a. Notice that we then have to explicitly add a step showing that $C \neq P$, since the linear reasoning tactic will not infer this automatically: it only rewrites equalities, inequalities and betweenness claims, so we must feed it the necessary facts explicitly.

This pattern of using the linear reasoning tactic with very few assumptions and lemmas, and then adding more in until the goal is solved, was our typical use-case of the tactic. We benefit from the fact that the tactic is a decision procedure, and the problems

we throw at it are normally sufficiently constrained that a yes/no answer is delivered promptly. As such, the tactic can be used to explore ideas as well as verify steps that are known in advance to be valid (see §11.3.1 for further discussion).

7.2.2 Covering

Our next theorem shows that there are at most two half-planes to each plane. This theorem is lifted from an analogous theorem on our intermediate type, but the basic details are again a strengthening of Pasch's axiom.

We need to prove that of three points A, B and C in a plane α containing a line a, it cannot be the case that A, B and C are on mutually distinct sides of a. In terms of our definition (7.2), this amounts to showing that a cannot simultaneously intersect between the pair of points A and B, the points A and C and the points B and C.

Thus, if ABC is a triangle, we are being asked to refute the possibility that the line a intersects all three sides. This fact would be immediate if the conclusion of Pasch's axiom was rendered with the exclusive-or.

This might well have been the case in the first edition of the *Grundlagen*. Hilbert uses an "either...or" for the axiom (and the analogous construction in the German edition). By the tenth edition, the "either" has disappeared, and now, Hilbert makes the explicit claim that the inclusive case can be refuted. It is clear, then, that he intends the weak inclusive-or in the axiom, and expects the inclusive case to be proved impossible.

Bernays thought the mere claim of a proof's existence was insufficient. In Supplement I to the text, he gives the proof in full:

> It behooves one to deduce the proof by means of THEOREM 4. It can be carried out as follows: If the line a met the segments BC, CA, AB at the points D, E, F then these points would be distinct. By THEOREM 4 one of these points would lie between the other two.
>
> If, say, D lay between E and F, then an application of Axiom II, 4 to the triangle AEF and the line BC would show that this line would have to pass through a point of the segment AE or AF. In both cases a contradiction of Axiom II, 3 or Axiom I, 2 would result.
>
> [42, p. 200]

This is an indirect proof, effectively based on an impossible diagram. The key inference is in the second paragraph; that is, if A, C and E are collinear, then we can use

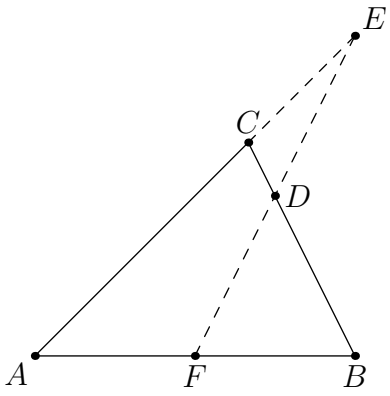


Figure 7.2: Supplement I

Pasch's Axiom to conclude that C must lie between A and E, contradicting our assumptions. The situation is shown in Figure 7.2, where we see how this step corresponds precisely to a use of the outer version of the Pasch axiom (5.2).

The verification is shown in Figure 7.3. Our incidence discoverer from Chapter 4 helps keep the proof steps almost one to one with the prose. We start by concluding as Bernays does that the points D, E and F are distinct and then apply THEOREM 4 to show that one of the points lies between the other two.

Bernays next makes a without-loss-of-generality assumption. We capture this with a subproof. There is some ugly repetition here with our `assume` steps, but after this comes the two key inferences. Note that we use the outer form of the Pasch Axiom (5.2), but, unlike Bernays, we leave out any mention of (I, 2). If we had to be consistently fussy in citing this axiom, it would have already appeared in the first step of the proof when showing that D, E and F are distinct. We leave it to implicit automation with our `obviously` step.

Finally, we can strengthen this supplement by removing the assumption that ABC forms a triangle. When A, B and C are collinear, we have a linear problem, and our incidence discoverer and linear reasoning tactic can do all the work in just four steps. With this case considered, we can give Bernays' supplement in a very general form:

$$\begin{aligned} \vdash & \neg\texttt{on\_line}\ A\ a \wedge \neg\texttt{on\_line}\ B\ a \wedge \neg\texttt{on\_line}\ C\ a \\ & \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a \\ & \Longrightarrow \neg\texttt{between}\ A\ D\ B \vee \neg\texttt{between}\ A\ E\ C \vee \neg\texttt{between}\ B\ F\ C. \end{aligned} \tag{7.3}$$

We have now strengthened Pasch's axiom (II, 4) in two ways: we have removed the assumption that A, B and C is a triangle, and we have removed the inclusive-or from

the conclusion. Respectively, these two facts tell us that Hilbert's same-side relation for half-planes is transitive, and that there are at most two half-planes on any given plane.

```
assume ¬(∃a. on_line A a ∧ on_line B a ∧ on_line C a)                    0
        on_line D a ∧ on_line E a ∧ on_line F a                          1
assume between A D B ∧ between A E C ∧ between B F C                     2
obviously (by_ncols ∘ conjuncts) have D ≠ E ∧ D ≠ F ∧ E ≠ F from 0, 2
hence between D E F ∨ between D F E ∨ between F D E from 1 by (THEOREM 4) 3
have ∀A'. ∀B'. ∀C'. ∀D'. ∀E'. ∀F'. ¬(∃a. on_line A' a ∧ on_line B' a ∧ on_line C' a)
    between A' F' B' ∧ between A' E' C' ∧ between B' D' C'
     ⟹ ¬between E' D' F'
proof:
    fix A', B', C', D', E', F'
    assume ¬(∃a. on_line A' a ∧ on_line B' a ∧ on_line C' a)             4
    assume between A' F' B' ∧ between A' E' C'
        ∧ between B' D' C' ∧ between E' D' F'                            5
    obviously (by_ncols ∘ conjuncts) consider G
        such that between A' G E' ∧ between B' D' G
        by (5.2), (II, 1) from 4, 5
    obviously (by_eqs ∘ conjuncts) qed from 4, 5 by (II, 3)
qed from 0, 2, 3 by (II, 1)
```

Figure 7.3: Proof for Supplement I

7.3 THEOREM 8

It is not enough to say that at most two half-planes cover a plane. We must also show that there are *at least* two half-planes in each plane. To that end, suppose we have a plane α and a line a in α. We find a point A on a, and a planar point B off the line a. Then, with Axiom II, 2, we can extend the segment BA through the line a to find a point C on the other side. The points B and C will then lie in distinct half-planes.

Note that this proof requires that the point B always exists, which is a consequence of Theorem 3.1 from Chapter 3. We noted at the time that this theorem was originally an axiom, but was later factored out. Hilbert does not explicitly say how the theorem is to be recovered, and as we suggested at the time, we do not believe the matter to be completely trivial.

Our final rendition of THEOREM 8 is a theorem lifted from our intermediate type. It relies on two lifted functions. The function `line_of_half_plane` returns the boundary of a given half-plane, while the function `half_plane_contains` is the incidence relation for half-planes. We give some additional theorems to govern these concepts in Figure 7.4, most of which are referenced in our account of the Polygonal Jordan Curve Theorem in Chapters 10 and 11. Meanwhile, THEOREM 8 is verified as:

$$\begin{aligned}
&\vdash (\forall P.\ \texttt{on_line}\ P\ a \implies \texttt{on_plane}\ P\ \alpha) \\
&\quad \implies \exists hp.\ \exists hq.\ hp \neq hq \\
&\qquad \wedge a = \texttt{line_of_half_plane}\ hp \wedge a = \texttt{line_of_half_plane}\ hq \\
&\qquad \wedge (\forall P.\ \texttt{on_plane}\ P\ \alpha \\
&\qquad\quad \iff \texttt{on_line}\ P\ a \vee \texttt{half_plane_contains}\ hp\ P \vee \texttt{on_half_plane}\ hq\ P).
\end{aligned} \tag{7.4}$$

Note that we have broken convention with the name for our incidence relation, using

$$\texttt{half\_plane\_contains} : \texttt{half\_plane} \to \texttt{point} \to \texttt{bool}$$

and not

$$\texttt{on\_half\_plane} : \texttt{half\_plane} \to \texttt{point} \to \texttt{bool}.$$

We can understand why this is necessary by considering exactly what we would have to say is well-defined to obtain this relation. It is a set of points which are incident with a given representative of half-planes from our intermediate type. Now as predicates, point-sets are functions of type $\texttt{point} \to \texttt{bool}$, and it is a function of this form which we must verify as well-defined. We obtain the predicate by partial application of an intermediate incidence relation `half_plane_intermediate_contains`:

$$\begin{aligned}
&\vdash \texttt{equiv_intermediate_half_plane}\ x\ y \\
&\qquad\qquad \implies \texttt{half_plane_intermediate_contains}\ x \\
&\qquad\qquad\qquad\qquad = \texttt{on_half_plane_intermediate_contains}\ y.
\end{aligned}$$

Consequently, the type `point` must appear last in the type of our half-plane incidence relation.

$$\begin{aligned}
&\vdash (\forall P.\ \texttt{on_line}\ P\ a \implies \texttt{on_plane}\ P\ \alpha) \\
&\quad \implies \exists hp\ \exists hq.\ hp \neq hq \\
&\qquad \wedge a = \texttt{line_of_half_plane}\ hp \wedge a = \texttt{line_of_half_plane}\ hq \\
&\qquad \wedge (\forall P.\ \texttt{on_plane}\ P\ \alpha \\
&\qquad\quad \iff \texttt{on_line}\ P\ a \vee \texttt{half_plane_contains}\ hp\ P \vee \texttt{on_half_plane}\ hq\ P)
\end{aligned}$$

$$\begin{aligned}
&\vdash \texttt{half_plane_contains}\ hp\ P \wedge \texttt{on_plane}\ P\ \alpha \\
&\quad \wedge (\forall R.\ \texttt{on_line}\ R\ (\texttt{line_of_half_plane}\ hp) \implies \texttt{on_plane}\ R\ \alpha) \\
&\quad \implies \texttt{half_plane_contains}\ hp\ Q \implies \texttt{on_plane}\ Q\ \alpha
\end{aligned} \tag{7.5}$$

$$\vdash \texttt{half\_plane\_contains}\ hp\ P \implies \neg \texttt{on\_line}\ P\ (\texttt{line\_of\_half\_plane}\ hp) \tag{7.6}$$

$$\begin{aligned}
&\vdash (\forall R.\ \texttt{half_plane_contains}\ hp\ R \implies \texttt{on_plane}\ R\ \alpha) \wedge \texttt{on_half_plane}\ hp\ P \\
&\quad \implies (\texttt{half_plane_contains}\ hp\ Q \\
&\qquad \iff \neg(\exists R.\texttt{on_line}\ R\ (\texttt{line_of_half_plane}\ hp) \wedge \texttt{between}\ P\ R\ Q) \\
&\qquad\quad \wedge \texttt{on_plane}\ Q\ \alpha \wedge \neg \texttt{on_line}\ Q\ (\texttt{line_of_half_plane hp}))
\end{aligned} \tag{7.7}$$

$$\begin{aligned}
&\vdash \texttt{on_line}\ P\ (\texttt{line_of_half_plane}\ hp) \wedge \texttt{half_plane_contains}\ hp\ Q \\
&\quad \implies \texttt{between}\ P\ Q\ R \vee \texttt{between}\ P\ R\ Q \implies \texttt{half_plane_contains}\ hp
\end{aligned} \tag{7.8}$$

$$\begin{aligned}
&\vdash \texttt{half_plane_contains}\ hp\ P \wedge \texttt{on_half_plane}\ hp\ R \\
&\quad \implies \texttt{between}\ P\ Q\ R \implies \texttt{half_plane_contains}\ hp\ Q
\end{aligned} \tag{7.9}$$

Figure 7.4: Theorems for half-Planes

7.4 Conclusion

All preliminaries needed for the Polygonal Jordan Curve Theorem have now been dealt with. This chapter completes the necessary theory, showing how to use the quotienting facilities of HOL Light to formalise the all important notion of half-planes. As we have seen, the fact that these objects exist and cover their planes can be understood as a strengthening of Pasch's Axiom.

We just note that one direction in which this axiom is weakened was singled out by Bernays as needing an explicit proof. We feel the same way about Theorem 3.1, which is also needed to prove THEOREM 8. Both supplementary proofs were unnecessary in the first edition, when the results were asserted axiomatically. But now that they have been factored out, one should be careful to derive them.

This omission is nothing compared to the chasm which follows. Hilbert gives no indication of how to prove THEOREM 9. A discussion of this theorem and its verification take up the next four chapters.

Chapter 8

Background to the Jordan Curve Theorem for Polygons

THEOREM 9 in the 10th edition of the *Grundlagen der Geometrie* is the focus of this thesis. It may only be a special case of the full Jordan Curve Theorem, but in the setting of ordered geometry, it is quite involved. In this brief chapter, we will draw together a number of historical threads and characters connected to the theorem and the *Grundlagen der Geometrie*, and we will look at the first published proof attempt by Veblen. This proof is severely inadequate, but there is enough to salvage from the approach to give a correct proof whose verification we leave to Chapters 10 and 11.

8.1 Relationship with the Full Jordan Curve Theorem

The full Jordan Curve Theorem effectively says that, when it comes to closed curves that do not self-intersect (*simple closed curves*), we are justified in our use of the expressions "inside the curve" and "outside the curve". The idea that mathematicians should even bother justifying this appears relatively late in the history of mathematics. In fact, it had to wait until the 19th century and the rigorous reformulations of analysis which reduced the unclear notions governing the continuum to precisely defined formulas involving the now standard ε and δ inequalities. Bolzano, who is credited along with Cauchy for spotting the reformulation, provided his own rigorous definitions for closed, continuous curves and what it means for curves to enclose points, so that he was then able to recommend the following for rigorous proof:

> "If a closed line lies in a plane and if by means of a connected line one

> joins a point of the plane which is enclosed within the closed line with a point of the plane which is not enclosed within it, then the connected line must cut the closed line."
>
> [48, p. 285]

This is a significant half of the Jordan Curve Theorem, and, reading the terms "closed lines" intuitively, it seems so blindingly obvious that one might think it perverse to demand a proof. However, the rigorous definitions which Bolzano had in mind to replace "closed line" are not so immediately intuitive, but appeal to very general and abstract topological properties. The first proof that these abstract properties preserve intuitive properties such as Bolzano's conjecture was given in Jordan's 1887 *Cours d'analyse* [49]. To this day, the proof is regarded as challenging, and one possible reason for the complexity is that the rigorous formulations are so general that they admit weird pathologies, or as Poincaré colourfully called them, *monsters*. Some of these monsters, such as closed curves enclosing finite area but having infinite length, immediately thwart a number of obvious proof strategies.

Relevant to this chapter is the case against Jordan's proof: common folklore says that it is invalid, and that the first correct proof was provided by Veblen in 1905 [101]. Both Veblen and folklore point out that Jordan had to assume the polygonal case, and argue that he should have proven this as a lemma. Hales, on the other hand, has formally verified the theorem in HOL Light, and put together a strong defence of Jordan [31], and of a basically elegant and correct proof that has been unfairly neglected. The polygonal case is supposedly completely trivial.

Not so, according to Feferman. In his paper concerning the aforementioned *monsters* [23], he repeats the folklore that Veblen gave the first correct proof, and claims that even the polygonal case is "devilishly difficult to prove."

There is already controversy with Hilbert's 1899 edition of the *Grundlagen der Geometrie*. There, Hilbert gave a formulation of the polygonal case as THEOREM 6, but, as with the five preceding theorems, he did not give a proof. Instead, he assures us that with the aid of his theorem for the existence of half-planes (THEOREM 5 of that edition), one can obtain the proof "without much difficulty." This certainly backs up the idea that the theorem is trivial. However, by the Ninth Edition, the clause had been deleted, with the edit noted as a "correction." The theorem still appears without proof, though Bernays, in a supplement to the main text, cites a detailed proof by Fiegl [24]. Note that Bernays does not *include* the proof, as he does in other cases, such as proving

that Pasch's axiom can be rendered without the inclusive-or (see §7.2.2). That would have taken more than a few supplementary remarks.

Now Veblen, one year before publishing his proof of the Jordan Curve Theorem, had developed an axiomatic foundation for geometry which was very close to Hilbert's own [100], and in his thesis, he expends a great deal more effort developing a theory of order than did Hilbert. This explains why Veblen's doctoral supervisor, E. H. Moore, was contributing proofs to later editions of Hilbert's text (see §5.3). It also explains why, in Veblen's 1905 proof of the full Jordan Curve Theorem, he thought it necessary to cite a proof from his doctoral thesis showing that the Jordan Curve Theorem holds for the even more trivial case of *triangles*. Plausibly, Veblen's criticism of Jordan can be explained by his particular standards of rigour and the context of an axiomatic theory of geometry.

Another aspect we should consider is the level of generality that Veblen was attempting. He gave a proof of the full theorem in the context of ordered geometry with the addition of one topological axiom. As such, the proof was not supposed to require any assumptions about the existence of a metric. Unfortunately, as Hales points out in his defence of Jordan [31], such generality was refuted ten years later by R. L. Moore[1], who had been a student of Veblen's. Moore showed that in Veblen's setting, all planes are homeomorphic to the Euclidean plane [74], and thus his axioms always describe a metrisable space.

But what about the polygonal case? In his doctoral thesis, Veblen gave a detailed, standalone proof of this theorem without using the topological axiom. The theorem, then, is plausibly still very general. So one question is: given its generality, is it still *trivial*? We suggest not. As we discuss in §8.4, a correct proof seems to have eluded Veblen himself.

8.2 Generality of the Polygonal Case

We can discuss the generality of ordered geometry by considering the two proofs of the polygonal theorem from the book *What Is Mathematics?* [81]. Hales mentions these to highlight the triviality of the polygonal case. The first of the proofs is the so-called "plumb-line" proof. We begin with a simple polygon, pick an arbitrary direction which is not parallel to any of its sides, and then for any point, we cast a ray in that direction.

[1]Not to be confused with Veblen's supervisor E. H. Moore.

By considering the number of times the ray crosses the polygon, and how this number changes as we move around the plane, we can prove the polygonal case.

This is a non-starter. The notion of *direction* should either be formalised in terms of angles, giving a compass direction, or in terms of equivalence classes of parallel lines. In Group II, we cannot say that parallel lines even *exist*. This matter is settled in Group IV, where the parallel axiom appears. Theorems for angle construction and congruence are also unavailable, based on axioms only available in Group III. We also have no formulations of what it means to "move" along a ray. The sort of motion being considered is presumably *continuous* motion, and indeed, if we look at Tverberg's proof of the theorem [97], he essentially gives the same argument in rigorous form, and appeals directly to continuity. But continuity does not appear in Hilbert's text until Group V.

The second proof from *What Is Mathematics?* only states an approach, and does not provide a complete proof. It effectively says we can prove the Jordan Curve Theorem for polygons by computing winding numbers for the polygon. A naïve formulation of this argument in terms of angles and continuous motion will be well outside the scope of Hilbert's first two groups of axioms, for the same reasons as above.

To reiterate, the problem we have with traditional proofs of the Polygonal Jordan Curve Theorem is that our axioms are too weak to formulate them. Another way to put this is to say that the version of the theorem we are attempting to prove is more general than the traditional versions, and must apply to *any* ordered geometry. A visual way to bring this point home is to note that by "simple polygons", we are actually including the boundaries of all possible *mazes* which do not contain loops, such as the one shown in Figure 8.1. Moreover, we are allowing the corridors of these mazes to be infinitesimally narrow, since we do not rule out non-Archimedean geometries at this stage.

Furthermore, we are tasked with navigating these mazes without being able to measure any distances. We cannot orient ourselves, rotate or compare directions. We cannot consider continuous motion. And we know nothing about the existence of parallel lines. In other words, we are navigating without a ruler, without a compass, and without being able to run a path parallel to a wall. We suspect these constraints will eliminate most trivial proofs.

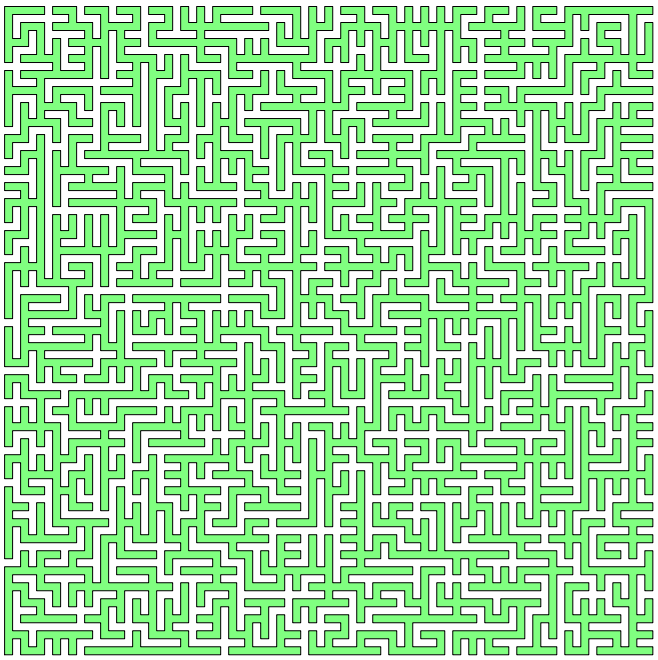


Figure 8.1: A simple polygon

8.3 Polygonal Case: Formulation

The full Jordan Curve theorem applies to arbitrary simple closed curves, and characterises the interior and exterior in terms of path-connectedness. The polygonal version of the Jordan Curve Theorem replaces "simple closed curve" with "simple polygon", and characterises the two regions in terms of *polygonal*-path connectedness. That is, the interior and exterior of the polygon are the largest sets all of whose points can be joined by polygonal paths.

The three primitives at Hilbert's disposal, namely two incidence relations and a betweenness relation, are sufficient to formulate the notion of polygons, interiors and exteriors. Having already defined a segment as an unordered pair of points, Hilbert defines a *polygonal segment*[2] as follows:

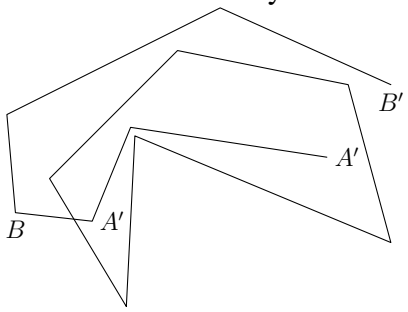
> DEFINITION. A set of segments AB, BC, CD, …, KL is called a *polygonal segment* that connects the points A and L. Such a polygonal segment will also be briefly denoted by $ABCD\ldots KL$. The points inside the segments AB, BC, CD, …, KL as well as the points A, B, C, D, …, K, L are collectively called the *points of the polygonal segment*.
>
> [42, p. 8]

[2] Veblen calls these *broken lines*.

Polygons are then polygonal segments where the points A and L coincide. For polygons, we refer to each of the individual segments AB, BC, …, KL as a *side* of the polygon. Finally, Hilbert defines *simple polygons*:

> "If the vertices of a polygon are all distinct, none of them falls on a side and no two of its nonadjacent sides have a point in common, the polygon is called *simple*."
>
> [42, p. 9]

Like the rays and half-plane theorems, the Polygonal Jordan Curve Theorem describes a partitioning of a space into two "connected" regions, where connectedness is cashed out in terms of a suitable relation. Here, we are told that the polygon partitions all other points in the plane as follows:

> THEOREM 9. Every single polygon lying in a plane α separates the points of the plane α that are not on the polygonal segment of the polygon into two regions, the *interior* and the *exterior*, with the following property: If A is a point of the interior **(an interior point)** and B is a point of the exterior **(an exterior point)** then every polygonal segment that lies in α and joins A with B has at least one point in common with the polygon. On the other hand if A, A' are two points of the interior and B, B' are two points of the exterior then there exist polygonal segments in α which join A with A' and others which join B with B', none of which have any point in common with the polygon. By suitable labelling of the two regions there exist lines in α that always lie entirely in the exterior of the polygon. However, there are no lines that lie entirely in the interior of the polygon.
>
> 
>
>
> [42, p. 9]

8.4 Veblen's Proof

In his 1903 doctoral thesis [100], Veblen set out a basic set of axioms for Euclidean Geometry. His incidence and order axioms are very similar to Hilbert's own, and it should not be difficult to verify their equivalence, but his attention to the elementary

theorems of ordered geometry is much more thorough. He proves forty theorems while Hilbert only proves ten, and he attempts a complete proof for the Polygonal Jordan Curve Theorem.

Veblen does not explicitly define the interior or exterior of a polygon. Like most other proofs, he tries to show that the two regions exist implicitly by dividing the result into two claims: the first states that a simple polygon divides its plane into at least two regions; the second states that the polygon divides its plane into at most two regions. Both assertions can be formalised in terms of polygonal segments:

1. there are at least two points in the plane not on the polygon which cannot be connected by a polygonal segment without crossing the polygon;

2. of any three points in the plane not on the polygon, at least two of them can be connected by a polygonal segment without crossing the polygon.

Veblen has a two page proof for this and it is one of the most detailed in his thesis, but it is far from persuasive. According to Reeken and Kanovei [51], the proof was deemed "inconclusive" by Hahn, while Guggenheimer claims, citing Lennes and Hahn, that Veblen's proof assumes that the polygon can be triangulated and is thus only valid for convex polygons [27]. We could not find this criticism in Lennes [59], and indeed, we think Guggenheimer is mistaken here. He may have been misled by the fact that Veblen's proof is based on finding a sequence of triangles whose vertices are shared with the polygon. But this sequence is no *triangulation*.

We were initially satisfied by Veblen's proof, and so we attempted to verify it. Eventually, as the sceptical Guggenheimer and Hahn might have anticipated, we hit an obstacle, and we had to give up. In the next few sections, we will suggest where Veblen's proof goes wrong.

8.4.1 Veblen's Lemma

The first half of Veblen's proof, showing that there are at least two points not on a polygon which cannot be connected, is given as the corollary to a very general result about polygons. In stating this result, he uses the term "multiple points", which are points, should they exist, where a polygon self-intersects:

> "*If a side of a polygon q intersects a side of a polygon p_n in a single point O not a multiple point of p_n or q, then p_n and q, whether simple or not, have at least one other point in common.*"

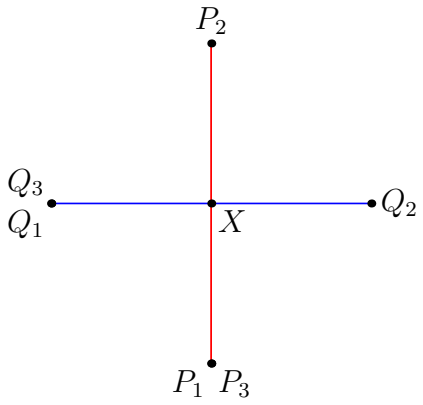


Figure 8.2: Degenerate polygons intersecting at a multiple point

[100, p. 365]

From this, we can obtain the first half of the Polygonal Jordan Curve Theorem, as we explain in §10.7, but this is already a very general result, and one we draw special attention to when we verify it in Chapter 10. We can see that it holds even for degenerate polygons. Consider the example in Figure 8.2. Here, we have two polygons $P_1P_2P_3$ and $Q_1Q_2Q_3$. The points of these polygons are obviously collinear and so cannot divide the plane into multiple regions. But neither are they a counterexample to Veblen's claim, since their point of intersection X is a multiple point of both, lying simultaneously on the segments P_1P_2 and P_2P_3, and also lying on the segments Q_1Q_2 and Q_2Q_3.

Though we have verified this result from Hilbert's axioms (see §10.6.2), we gave up trying to reproduce Veblen's argument. We do not have a counterexample, as such, since we do not think Veblen's proof is sufficiently detailed to say exactly where it fails. Instead, we have tried to illustrate the difficulties we faced with the example in Figure 8.3. This example shows a simple maze polygon p_n being intersected at a point O by another polygon q shown in red. The goal is to identify one of the other seven points of intersection. Our labelling in the diagram is consistent with the set-up to Veblen's proof:

> If $n = 3$ (q having any number of sides, m) the theorem reduces to [the case for triangles]. We assume without loss of generality that no three vertices P_{i-1}, P_i, P_{i+1} are collinear and prove the lemma for every n by reducing to the case $n = 3$. Let p_n have n vertices with the notation such that the side P_1P_2 meets q in the side $Q_1'Q_2'$ where the segment $Q_2'O$ contains no interior point of the triangle $P_1P_2P_3$.

[100, p. 365]

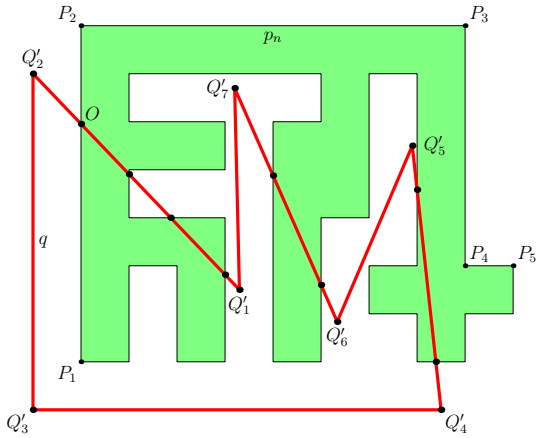


Figure 8.3: Intersections on a simple maze

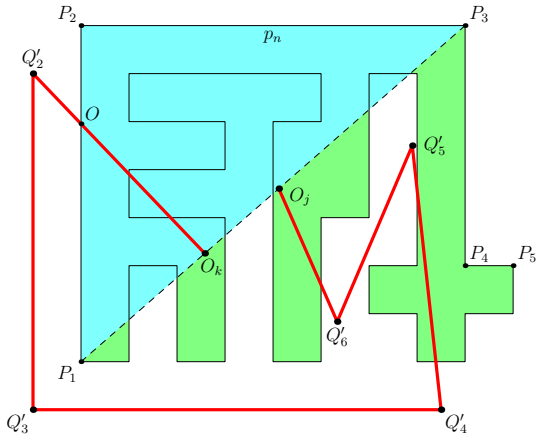


Figure 8.4: A chosen subset of q: $k = 2$ and $j = 6$

Veblen's basic strategy is to consider each of the triangles $P_1P_2P_3$, $P_1P_3P_4$, $P_1P_4P_5$, $P_1P_5P_6$, ... Of these triangles, all but the first and last share exactly one side with the polygon p_n, with the other sides being diagonals of the polygon. Veblen tries to show that if q intersects one of these triangles in one diagonal, then it intersects the next triangle. In this way, intersections can be found, one after the other, down the list of triangles, until we eventually find a second point of intersection with the polygon p_n.

8.4.2 Finding a subset of q

In Figure 8.4, we show a polygonal segment (or "broken line") which is a subset of the polygon q, with vertices $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$. This segment makes contact with the diagonal P_1P_3 exactly twice, once from the outside of the triangle $P_1P_2P_3$, and once from the inside. The polygonal segment always exists, and can be proven to intersect

P_1P_3 in just this way. To find it, we start from the segment $Q'_2Q'_3$ and progressively add neighbouring segments until we eventually reach a segment which intersects the line P_1P_3. As Veblen puts it:

> By the case $n = 3$, q meets the boundary of the triangle $P_1P_2P_3$ in at least one point other than O. If this point is on the broken line $P_1P_2P_3$ the lemma is verified. If not, q has at least one point on P_1P_3, and at least one of the segments $Q'_1Q'_2$, $Q'_2Q'_3$ has no point or end-point on P_1P_3. Let this segment be one segment of a broken line $Q_kQ_{k+1}\cdots Q_{j-1}Q_j$ of segments of q not meeting P_1P_3 but such that $Q_{k-1}Q_k$ and Q_jQ_{j+1} do each have a point or endpoint in common with P_1P_3 ($1 \leq k < j \leq m$; if $k = 1$, $Q_{k-1} = Q_m$; if $j = m$, $Q_{j+1} = Q_1$). If O_j is the point common to P_1P_3 and Q_jQ_{j+1} or Q_{j+1}, and O_k is the point common to P_1P_3 and $Q_{k-1}Q_k$ or Q_{k-1}, the broken line $O_kQ_kQ_{k+1}\cdots Q_{j-1}Q_jO_j$, has a point inside and also a point outside the triangle $P_1P_2P_3$ and cuts the broken line $P_1P_2P_3$ only once.
>
> [100, p. 365]

Now there is nothing particularly informative about Veblen's last remark. Notice that the segment $Q'_1Q'_2$ in Figure 8.3 also has a point inside and a point outside the triangle $P_1P_2P_3$. It also cuts the polygonal segment $P_1P_2P_3$ exactly once. Something is missing here. There must be some other property had by the polygonal segment $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$, in order for Veblen to get to his very next claim, that "it has a point inside and a point outside any triangle of which P_1P_3 is a side". The missing property is an important detail, because as we mentioned, part of the argument is supposed to be repeated down the list of triangles $\triangle P_1P_2P_3$, $\triangle P_1P_3P_4$, $\triangle P_1P_4P_5$, $\triangle P_1P_5P_6$, ... If we are going to repeat this part of the argument, we need to know that the missing property is an *invariant*.

For now, we just note that Veblen's claim certainly follows. We believe the crucial point is that, as we remarked earlier, the polygonal segment $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$ touches P_1P_3 exactly twice and from opposite sides. More precisely, we just note that the interior of O_kO is inside the triangle $P_1P_2P_3$,[3] and thus by the Jordan Curve Theorem applied to triangles, all points of the polygonal segment $O\cdots Q'_2Q'_3Q'_4Q'_5Q'_6O_j$ other than O_j must be outside the triangle. It is then possible to show that the points inside the segment O_kO are on the opposite side of the line P_1P_3 as the points inside the segment Q'_6O_j, and so must be in different regions of any triangle of which P_1P_3 is a side. Veblen's claim then follows: the polygonal segment $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$ has a

[3] For Veblen, this is a matter of definition. See §10.3.

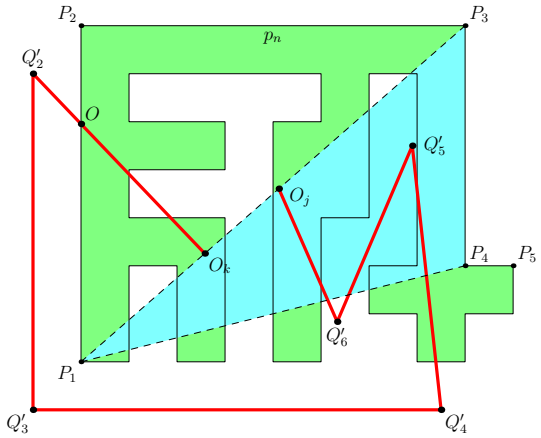


Figure 8.5: Intersections with $P_1P_3P_4$

point inside and a point outside any triangle of which P_1P_3 is a side. We just needed a bit of extra work to get there.

8.4.3 Veblen's Conclusion

> On this account if $P_1P_3P_4$ are not collinear, and obviously, if $P_1P_3P_4$ are collinear, q must meet either P_3P_4 or P_4 or P_4P_1. If q does not meet P_3P_4 or P_4, we proceed with $P_1P_4P_5$ as we did with $P_1P_3P_4$. Continuing this process, we either verify the lemma or come by $n-2$ steps to the triangle $P_1P_{n-1}P_n$ and find that q must intersect the broken line $P_{n-1}P_nP_1$, which also verifies the lemma.
>
> [100, p. 365]

This is Veblen's conclusion, and the situation described is illustrated in Figure 8.5. The first step follows by the Jordan Curve Theorem applied to triangles: we know that $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$ does not strictly *cut*[4] the line P_1P_3, so it must instead cut either P_1P_4 or P_3P_4. We can assume that q does not meet P_3P_4, and so we proceed with $P_1P_4P_5$. But Veblen is not clear on exactly how the argument repeats. The reason the first step follows in the above conclusion is because $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$ does not cut P_1P_3, but we have now assumed that it *does* cut P_1P_4, so we are not simply repeating the conclusion.

Notice that in this conclusion, Veblen has slipped to talking about the original polygon q rather than the polygonal segment $O_kQ'_2Q'_3Q'_4Q'_5Q'_6O_j$ which is a subset of q. Presumably then, we need to find another subset of q which has the same properties

[4]The use of phrases such as "strictly cut", among many other details, will be clarified in our verification. See Chapter 10.

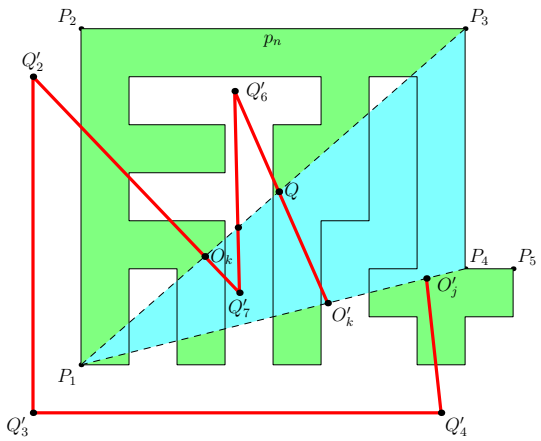


Figure 8.6: Desired subset of q?

as the original. And here we encounter the problem of detail mentioned at the end of §8.4.2: we are just not sure what the crucial properties of the subset *are*. Nevertheless, we tried earlier to fill in those details, and based on our attempt, we might assume that the desired polygonal segment for our example is the one shown in Figure 8.6, namely $O'_kQ'_6Q'_7Q'_2Q'_3Q'_4O'_j$. This polygonal segment can be found by starting at the point O_k and following Veblen's procedure as we did with the point O.

Repeating his argument, Veblen's next claim should be that the polygonal segment $O'_kO_kQ'_2Q'_3Q'_4O'_j$ has a point inside and outside any triangle of which P_1P_4 is a side. Again, this is certainly true. But following our earlier justification, we would establish the fact by noting that the polygonal segment touches P_1P_4 exactly twice, once from inside the triangle and once from outside. More formally, we observe that the points inside the segment O'_kQ lie inside the triangle $P_1P_3P_4$, while the points inside the segment $Q'_4O'_j$ lie outside the triangle.

In the previous case, we made the observation using Veblen's assertion that the polygonal segment $O'_kO_kQ'_2Q'_3Q'_4O'_j$ cuts the polygonal segment $P_1P_2P_3$ *exactly once*, but this is not the case for $O'_kQ'_6Q'_7Q'_2Q'_3Q'_4O'_j$ and $P_1P_3P_4$. Instead, the required observation is that it cuts $P_1P_3P_4$ an *odd* number of times, and so it has a non-empty suffix which lies entirely outside the polygon.

We therefore need a proof that the number of cuts must always be odd, but we cannot see anything in Veblen's argument that would require this. We presumably need additional lemmas about the parity of cuts on a triangle's side. But once we go down that path, we can concoct an argument which is conceptually much more straightforward. We give this argument in full in §10.1.

8.5 Final Remarks

There are at least three other published proofs of the Polygonal Jordan Curve Theorem from axioms equivalent to Hilbert's first two groups. Feigl's proof is cited by Bernays in the *Grundlagen der Geometrie* [24]. A second proof was provided by Main [61], whose doctoral work was supervised by the same Professor Goheen who wrote the foreword to the 10th edition of the *Grundlagen der Geometrie*. Both Feigl and Main's proof were written some decades after Veblen's proof, and both exploit a parity argument based on angles (defined as a pair of rays emanating from a single point).

Guggenheimer supplied a proof nine years after Main's [28]. His proof is conceptually more sophisticated than Feigl's and Main's, exploiting a theorem by Dehn [27] and showing that there exists a suitable homeomorphism from the plane to a half-plane which maps polygonal Jordan curves onto triangles. A consequence of this is that the theorem reduces to Pasch's Axiom II, 4.

We have contributed what amounts to a new proof by patching up the flaws in Veblen's. We sketch the details in §10.1.

Chapter 9

Formalising the Polygonal Jordan Curve Theorem

In this chapter, we introduce our verification of the Polygonal Jordan Curve Theorem, and we present its formalisation in full. We will set forth, unambiguously, *what* we have verified, and thus, much of this chapter can be read independently of the verification. The formal definitions and formalisations are not as simple as those in earlier chapters, and so they must be carefully checked. But once we are convinced that the formalisations correctly capture the statement of the theorem, we have an absolute guarantee that it is derivable from Hilbert's axioms.

9.1 Organisation

The verification is divided neatly into two parts, corresponding to the verification of two theorems, which we formalise in §9.3:

1. there are two points in the plane of a simple polygon such that any polygonal path connecting them must cross the simple polygon (verified in Chapter 10);
2. given three points in the plane of a simple polygon, there is a polygonal path connecting two of those points (verified in Chapter 11).

When we discuss the verifications, we will use the same basic structure. First, we shall give an overview of the theory structure, effectively amounting to a sketch proof of the theorem. From this, we shall give formalisations of any key concepts mentioned in the proof. In the case of the first part of the Polygonal Jordan Theorem, these are

the concepts of triangle interiors and exteriors and the concept of a "crossing" of a triangle by a polygonal path. For the second part of the theorem, we have concepts of "lines-of-sight" and polygon rotations.

These concepts are supported by a suite of lemmas. Consistent with the synthetic style of geometric proof, these lemmas generally break down into one of two forms. We have lemmas which introduce points and other objects in a geometrical configuration. We also have lemmas which allow us to infer properties of these configurations. The two sorts of lemmas are used in tandem: we build up complex figures using the "introduction" rules, and then use the other rules to satisfy the hypotheses of yet more "introduction" rules, and so on. Many of these lemmas are interesting in their own right, and we shall pick a few for closer examination.

We present some extracts of what we hope are interesting verifications, ones which highlight the benefits and drawbacks of our representations in terms of the resulting proof mechanics. We hope to use these extracts to convince the reader that we have stayed faithful to the synthetic style of proof, even as the verifications grow significantly in complexity. The complexity is easy enough to measure. The verifications from Chapter 5 take only a dozen or so steps. The verifications in the next few chapters frequently run to hundreds, and we suspect this really is due to the complexity of the synthetic reasoning involved.

9.2 Related Work

The Jordan Curve Theorem has some notoriety within the formal verification community, and has long been regarded as a major milestone, one which demonstrates the feasibility of formal verification on non-trivial mathematics problems. The MIZAR [19] community first began a verification in 1991, and completed the special case for polygons in 1996. The full proof was completed in 2005. In the same year, Hales completed an independent proof in HOL Light [32].

Both proofs use the special case for polygons as a lemma, though in a restricted form: in the case of the MIZAR proof, only polygons with edges parallel to axes are considered. In Hales' proof, the polygon is restricted to lie on a grid. Moreover, the formulations are algebraic rather than synthetic, and so are outside the scope of Hilbert's and Veblen's formulations.

In 2009, Dufourd [22] formally verified a constructive proof of the Discrete Jordan

Curve Theorem. This theorem characterises Jordan curves in terms of a "ring" of faces in a subdivision of a plane or sphere. This is a more practical achievement, since we can expect this sort of characterisation to be more useful to computer scientists trying to express and reason about topological properties computationally. However, our interest here is in Hilbert's formulation and a synthetic proof of the Jordan Curve Theorem in ordered geometry.

9.3 Formulation

> DEFINITION. A set of segments AB, BC, CD, …, KL is called a *polygonal segment* that connects the points A and L. Such a polygonal segment will also be briefly denoted by $ABCD\ldots KL$. The points inside the segments AB, BC, CD, …, KL as well as the points A, B, C, D, …, K, as well as the points A, B, C, D, …, K, L are collectively called the *points of the polygonal segment*.
>
> In addition, for a polygonal segment A, B, C, …, L, we shall call the segments AB, BC, CD, …, KL the *sides* of the polygonal segment.
>
> [42, p. 8]

This is Hilbert's first key definition, defining what Veblen calls *broken lines*, and what we shall prefer to call *polygonal paths*. Contrary to his conventions (see §3.2.2), Hilbert does not insist that the points involved here are distinct. Indeed, he adds an explicit distinctness clause when he defines simple polygons (see §9.3.2).

One thing which is clear to us is that Hilbert's notation is doing the heavy lifting in this definition. He uses it to take care of the otherwise clumsy constraint that all but two segments in the set share their endpoints with at least two other segments. The remaining two elements must share exactly one of their endpoints with another segment.

Had Hilbert given a proof of the polygonal Jordan Curve Theorem, he would have probably used his notation to do more heavy lifting, as Veblen did in his own proof. Veblen's argument ends up running over symbols and numerical subscripts in a way which seems to take us out of the world of synthetic geometry, and back into the sort of computational metatheory we saw in THEOREM 6 from Chapter 6.

> By the case $n = 3$, q meets the boundary of the triangle $P_1P_2P_3$ in at least one point other than O. If this point is on the broken line $P_1P_2P_3$ the lemma is verified. If not, q has at least one point on P_1P_3, and at least one of the segments $Q'_1Q'_2$, $Q'_2Q'_3$ has no point or end-point on P_1P_3. Let this segment be one segment of a broken line $Q_kQ_{k+1}\cdots Q_{j-1}Q_j$ of

> segments of q not meeting P_1P_3 but such that $Q_{k-1}Q_k$ and Q_jQ_{j+1} do each have a point or endpoint in common with P_1P_3 ($1 \leq k < j \leq m$; if $k = 1$, $Q_{k-1} = Q_m$; if $j = m$, $Q_{j+1} = Q_1$). If O_j is the point common to P_1P_3 and Q_jQ_{j+1} or Q_{j+1}, and O_k is the point common to P_1P_3 and $Q_{k-1}Q_k$ or Q_{k-1}, the broken line $O_kQ_kQ_{k+1}\cdots Q_{j-1}Q_jO_j$, has a point inside and also a point outside the triangle $P_1P_2P_3$ and cuts the broken line $P_1P_2P_3$ only once.
>
> [100, p. 365]

Looking at Veblen's proof attempt, one might expect the argument to be dominated by computations and lemmas about point sequences, and not by synthetic constructions of diagrams, but this is not entirely the case. Our verification shows that there are a *lot* of useful lemmas very similar to those we saw in Chapter 5, whose synthetic proofs obtain and then reason about properties of diagrams.

With the notation doing such heavy lifting, we will formalise it directly. We opted to represent polygonal paths as their finite sequence of points, from which the original polygonal paths can be recovered. With this definition, the clumsy constraint about segments sharing endpoints is handled implicitly. The heavy lifting will be carried out as computations on lists, which are available to us in the logic, ultimately being defined in terms of natural numbers. As we saw in Chapter 6, this means they are still ultimately represented by geometric figures.

Now the list library in HOL Light is slightly impoverished (at least compared with, say, that of Isabelle/HOL), and so our verification had to take a detour as we added new function definitions and theorems for lists. For instance, in order to recover the edges of a polygonal path, we must take adjacent pairs of the points in its vertex list. We can do this by following a standard pattern in functional programming, shown in Figure 9.1. The function `adjacent` zips all but the last element of a list with its tail. However, it is generally easier and requires much less unfolding to compute directly with the recursive specification of the function. We give this along with the definitions and specifications of other auxiliary functions in Figure 9.2.

With the function `adjacent`, we can define the points of a polygonal path. As per Hilbert's definition, these are the points of the vertex list and the points inside each individual segment (x,y). We can test for each using the list-membership predicate `mem`.

$$\begin{aligned} &\texttt{on\_polypath} : [\texttt{point}] \to \texttt{point} \to \texttt{bool} \\ \vdash_{def}\ &\texttt{on\_polypath}\ Ps\ P \iff \\ &\quad \texttt{mem}\ P\ Ps \vee \exists x.\ \exists y.\ \texttt{mem}\ (x,y)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ x\ P\ y. \end{aligned} \tag{9.1}$$

$$
\begin{aligned}
&\texttt{adjacent} : [\alpha] \to [(\alpha, \alpha)] \\
\vdash_{def} \ &\texttt{adjacent}\ [P_0, P_1, P_2, \ldots, P_n] \\
&\quad = \texttt{zip}\ (\texttt{butlast}\ [P_0, P_1, P_2, \ldots, P_n])\ (\texttt{tail}\ [P_0, P_1, P_2, \ldots, P_n]) \\
&\quad = \texttt{zip}\ \begin{bmatrix} P_0, & P_1, & P_2, & \ldots, & P_{n-1} \end{bmatrix} \\
&\qquad\qquad\ \ \begin{bmatrix} P_1, & P_2, & P_3, & \ldots, & P_n \end{bmatrix} \\
&\quad = [(P_0, P_1), (P_1, P_2), (P_2, P_3), \ldots, (P_{n-1}, P_n)]
\end{aligned}
$$

$$
\begin{aligned}
&\vdash \texttt{adjacent}\ [] = [] \\
&\vdash \texttt{adjacent}\ [x] = [] \\
&\vdash \texttt{adjacent}\ (x :: y :: xs) = (x, y) :: \texttt{adjacent}\ (y :: xs)
\end{aligned}
$$

Figure 9.1: Specifications for `adjacent`

Next, we need to formalise the notion of region as used in the Polygonal Jordan Curve Theorem. We shall follow our approach when formalising the notions of *half-plane* and *rays* (see §7.1.2), and understand these regions as equivalence classes under a suitable relation, namely one which requires there to be a polygonal path between two given points. When two points satisfy this relation, we shall say that they are *polygonal path-connected.*

$$
\begin{aligned}
\vdash_{def}\ &\texttt{polypath\_connected} : \texttt{plane} \to (\texttt{point} \to \texttt{bool}) \to \texttt{point} \to \texttt{point} \to \texttt{bool} \\
&\texttt{polypath\_connected}\ \alpha\ \mathit{figure}\ P\ Q \iff \\
&\quad \exists \mathit{path}.\ \mathit{path} \neq [] \\
&\qquad \wedge\ (\forall R.\ \texttt{mem}\ R\ \mathit{path} \implies \texttt{on\_plane}\ R\ \alpha) \\
&\qquad \wedge\ \texttt{head}\ \mathit{path} = P \wedge \texttt{last}\ \mathit{path} = Q \\
&\qquad \wedge\ \texttt{disjoint}\ (\texttt{on\_polypath}\ \mathit{path})\ \mathit{figure}.
\end{aligned}
$$

Note that, when formalised, this relation is defined on the set of all points in space, parameterised on a plane α and a figure. Instead of a plane parameter, we could have added constraints to the figure and path, such as requiring that they all lie in exactly one plane. The trade-offs are in the number of constraints in the definition compared with the number of constraints on later theorems. Here, we have opted for the simpler definition.

$$\texttt{head} : [\alpha] \to \alpha \qquad \vdash_{def} \texttt{head}\ (x :: xs) = x$$

$$\texttt{tail} : [\alpha] \to [\alpha]\dagger \qquad \vdash_{def} \texttt{tail}\ [] = [] \qquad \vdash_{def} \texttt{tail}\ (x :: xs) = xs$$

$$\texttt{length}\ [\alpha] \to \mathbb{N} \qquad \vdash_{def} \texttt{length}\ [] = 0 \qquad \vdash_{def} \texttt{length}\ (x :: xs) = \texttt{length}\ xs + 1$$

$$\texttt{butlast} : [\alpha] \to [\alpha]$$
$$\vdash_{def} \texttt{butlast}\ [] = []$$
$$\vdash_{def} \texttt{butlast}\ (x :: xs) = \texttt{if}\ xs = []\ \texttt{then}\ []\ \texttt{else}\ x :: (\texttt{butlast}\ xs)$$

$$\texttt{el} : \texttt{int} \to [\alpha] \to \alpha$$
$$\vdash_{def} \texttt{el}\ 0\ xs = \texttt{head}\ xs$$
$$\vdash_{def} \texttt{el}\ (\texttt{suc}\ n)\ xs = \texttt{el}\ n\ (\texttt{tail}\ xs)$$

$$\texttt{mem} : \alpha \to [\alpha] \to \texttt{bool} \qquad \vdash_{def} \texttt{mem}\ x\ [] = \bot \qquad \vdash_{def} \texttt{mem}\ x\ (y :: ys) = x = y \vee \texttt{mem}\ x\ ys$$

$$\texttt{all} : (\alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool} \qquad \vdash_{def} \texttt{all}\ p\ [] = \top \qquad \vdash_{def} \texttt{all}\ p\ (x :: xs) = p\ x \wedge \texttt{all}\ p\ xs$$

$$\texttt{pairwise} : (\alpha \to \alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool}$$
$$\vdash_{def} \texttt{pairwise}\ R\ [] = \top$$
$$\vdash_{def} \texttt{pairwise}\ R\ (x :: xs) = \texttt{all}\ (R\ x)\ xs \wedge \texttt{pairwise}\ R\ xs$$

$$\texttt{disjoint} : (\alpha \to \texttt{bool}) \to (\alpha \to \texttt{bool}) \to \texttt{bool}$$
$$\vdash_{def} \texttt{disjoint}\ S\ T = S \cap T = \emptyset$$

† This function is a more well-defined version of the existing function in HOL Light. The original version is undefined for the empty list.

Figure 9.2: List definitions and specifications

A figure here is represented by a predicate-set of all the points on the figure. Thus, the relation on points in the plane α which are polygonal path-connected with respect to a polygonal path *Ps* can be cleanly expressed by

$$\texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps).$$

This is obviously an equivalence relation.

9.3.1 Verifying Equivalence

The proof that we have an equivalence relation boils down entirely to properties of lists. To prove reflexivity, we use the one-element polygonal path (excluded in Hilbert's definitions). Our witness for symmetry is obtained by reversing the supplied polygonal path. Our witness for transitivity is obtained by appending the two supplied polygonal paths. To reason about the resulting lists, we first verified some extra simplification rules for our list functions:

$$\begin{aligned}
&\vdash xs \neq [] \wedge ys \neq [] \\
&\quad \Longrightarrow \texttt{adjacent}(xs + ys) \\
&\qquad = \texttt{adjacent}\ xs + (\texttt{last}\ xs,\ \texttt{hd}\ ys) + \texttt{adjacent}\ ys. \\
&\vdash n + 1 < \texttt{length}\ xs \\
&\quad \Longrightarrow \texttt{el}\ n\ (\texttt{adjacent}\ xs) = (\texttt{el}\ n\ xs, \texttt{el}\ (n+1)\ xs). \\
&\vdash \texttt{length}\ xs = \texttt{length}\ ys \\
&\quad \Longrightarrow \texttt{reverse}\ (\texttt{zip}\ xs\ ys) = \texttt{zip}\ (\texttt{reverse}\ xs)\ (\texttt{reverse}\ ys). \\
&\vdash \texttt{butlast}\ (\texttt{reverse}\ xs) = \texttt{reverse}\ (\texttt{tail}\ xs). \\
&\vdash \texttt{tail}\ (\texttt{reverse}\ xs) = \texttt{reverse}\ (\texttt{butlast}\ xs).
\end{aligned}$$

With these, we can verify three theorems showing that polygonal path-connectedness defines an equivalence relation. The domain of the relation is the set of points in the plane which are not points of the figure. The constraint appears as an assumption on reflexivity.

Obviously, we will need the three theorems *somewhere* in our later verification, though it turns out that they are not used very often. They can be seen, at least, as a sanity check on our notion of polygonal path-connectedness.

$$\vdash \texttt{on\_plane}\ P\ \alpha \wedge \neg \mathit{figure}\ P \Longrightarrow \texttt{polypath\_connected}\ \alpha\ \mathit{figure}\ P\ P.$$

$\vdash$`polypath_connected` α *figure P Q* $\Longrightarrow$ `polypath_connected` α *figure Q P*.

$\vdash$`polypath_connected` α *figure P Q* $\wedge$ `polypath_connected` α *figure Q R*

$\Longrightarrow$ `polypath_connected` α *figure P R*.

9.3.2 Polygons

Hilbert continues his definitions as follows:

> If the points A, B, C, D, ..., K, L all lie in a plane and the point A coincides with the point L then the polygonal path is called a *polygon* and is denoted as the polygon $ABCD\ldots K$. The segments AB, BC, CD, ..., KA are also called the *sides of the polygon*. The points A, B, C, D, ..., K are called the *vertices of the polygon.* Polygons of 3, 4, ..., n *vertices* are called *triangles*, *quadrilaterals*, ..., *n-gons*.
>
> [42, pp. 8–9]

The term "side" is ambiguous between the segments of a polygon and the half-planes of a given line. As such, we shall refer to the segments defining both a polygonal path and a polygon as *edges*, and reserve *side* for half-planes. These definitions will help us orientate ourselves within the familiar world of geometry, but we do not believe they correspond to any useful abstractions for verification. As such, we shall only use them informally to explain parts of the verification. The important definition for the verification is the one for *simple polygons*:

> "DEFINITION. If the vertices of a polygon are all distinct, none of them falls on [an edge] and no two of its nonadjacent [edges] have a point in common, the polygon is called *simple*."
>
> [42, p. 9]

Combining this with the definition of polygon gives us the formalisation in Figure 9.3. In this definition, we have introduced the polygon's plane as a parameter, but we are aware that this could be refactored. Any simple polygon uniquely determines the plane on which it lies, and so it might have been more elegant to hide the plane witness by an existential in the body of the definition. A function could then be defined to extract the unique witness from the list *Ps* when *Ps* is known to be a simple polygon.

The definition would still be rather unwieldy. First, we have a "magic" number 3,[1] which is needed to rule out the degenerate case of a point polygon $[P,P]$ which slips

[1] The number 4 would do just as well!

$$
\begin{aligned}
&\texttt{simple\_polygon} : \texttt{plane} \to [\texttt{point}] \to \texttt{bool} \\
&\vdash_{def} \texttt{simple\_polygon}\ \alpha\ Ps \iff \\
&\quad 3 \leq \texttt{length}\ Ps \\
&\quad \wedge \texttt{head}\ ps = \texttt{last}\ Ps \\
&\quad \wedge (\forall P.\ \texttt{mem}\ P\ Ps \implies \texttt{on\_plane}\ P\ \alpha) \\
&\quad \wedge \texttt{pairwise}\ (\neq)\ (\texttt{butlast}\ Ps) \\
&\quad \wedge \neg(\exists P.\ \exists Q.\ \exists X.\ \texttt{mem}\ X\ Ps \wedge \texttt{mem}\ (P,Q)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ P\ X\ Q) \\
&\quad \wedge \texttt{pairwise}\ (\lambda(P,Q)\ (P',Q').\ \\
&\qquad \neg(\exists X.\ \texttt{between}\ P\ X\ P' \wedge \texttt{between}\ Q\ X\ Q')\ (\texttt{adjacent}\ Ps)).
\end{aligned}
\tag{9.2}
$$

Figure 9.3: Formalisation of Simple Polygons

past Hilbert's constraints. But the real complexity is in the last three clauses which define the polygon as *simple*.

Given the unwieldiness of this formalisation, we must be wary of subtle mistakes. These could lead either to some figures being classed as simple polygons when they should not be, or some figures not being classed as simple polygons when they should. The first sort of error must show up in the verification. The second sort of error is more insidious, since it will only cause our verification to become all too easy. In the worst case, the definition will be unsatisfiable and all theorems of simple polygons will become trivial. This might happen if, say, we had removed the use of `butlast` above.

Of particular concern is the behaviour of the function `pairwise`. One might think that `pairwise` R would check whether the relation R holds across all pairs of elements drawn from its argument list, which would be the case if `pairwise` were equivalent to `all` $\circ$ `fmap2` R for the usual list monad. If this were the case, the definition would be unsatisfiable, since it is always possible to draw some pair (P,P) from a non-empty list, and this pair cannot satisfy $(\neq)$. But in fact, `pairwise` only checks half the pairs, and so can be satisfied even when the supplied relation is irreflexive (such as $(\neq)$ above). It even holds for anti-symmetric relations. Consider `pairwise` $(<)$ $[1,2,3,4]$:

$$
\begin{aligned}
\vdash \texttt{pairwise}\ (<)\ [1,2,3,4] = 1 < 2 < 3 < 4 \\
\wedge\ 2 < 3 < 4
\end{aligned}
$$

$$\begin{aligned} &\wedge\ 3 < 4 \\ &= \top. \end{aligned}$$

For now, inspection is the best way to inspire confidence in the definition, but there are two other small reassurances. Firstly, the definition does not appear until the final hurdles of our formal verification. Our verification of Veblen's lemma, for instance, makes no reference to simple polygons. Thus, there is plenty of verified theory which can be understood independently of the above definition. Secondly, we have the following simple sanity check, verifying that a triangle is a simple polygon.

$$\begin{aligned} \vdash \neg(\exists a.\ &\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \\ &\Longrightarrow \texttt{simple\_polygon}\ \alpha\ [A,B,C,A]. \end{aligned}$$

Note that for Chapters 10 and 11, we shall elide all terms involving `on_plane`, and thus present this verification as if from planar rather than spatial axioms. This is merely for clarity. Had we been working from planar axioms, all of these terms would be absent, since they serve only to relativise formulas to a single plane (see §3.2.5 for some discussion).

9.3.3 Goal Theorems

In §9.3.1, we said that we would understand the regions defined by the Polygonal Jordan Curve Theorem as equivalence classes under polygonal path-connectedness. If we were to follow the style of Chapter 7, we would quotient an appropriate data-type under this relation and regions would be abstract.

We shall not do this, however. We are not planning as of now to build on the Polygonal Jordan Curve Theorem, and so the abstraction can wait. We hope, though, that some of the example verifications in the next two chapters show that there *was* a pay-off when talking abstractly of half-planes, while there was no need to talk abstractly of rays. This gives us some perspective on both Hilbert and Veblen's remarks that the Polygonal Jordan Curve Theorem is principally founded on the theory of half-planes.

Abstractly then, the Polygonal Jordan Curve Theorem tells us that a simple polygon separates the plane into exactly two regions, but we will talk concretely in terms of the underlying representation and polygonal path-connectedness. We say firstly that there are at least two points in the plane and not on the polygon which are not polygonal

path-connected. Secondly, we say that of any three points in the plane and not on the polygon, at least two of them can be polygonal path-connected.

There is a final claim: one of the regions is unbounded. This theorem does not appear in Veblen's thesis, and we have left its verification to future work. Hilbert formulates the claim by saying that exactly one of the regions contains straight lines. Since we are avoiding talk of regions directly, we shall formulate it as follows:

- there exists a line in the plane of a polygonal path which does not intersect the polygonal path;
- given two lines a and b in the plane of a polygonal path which does not intersect the polygonal path, all points of a are polygonal path-connected to all points of b with respect to the polygonal path.

In the first part, the rough idea is that there is at least one region containing a straight line, while in the second, that there is at most one region containing a straight line. The formulation needs some discussion.

We have dropped the unnecessary condition that the polygonal path is a simple polygon from both parts. We have also dropped any mention of polygonal path-connectedness from the first part. A condition of polygonal path connectedness appears in the second part, and can be obtained for the first part simply by setting a and b to the first part's witness.

With the breakdown considered, we turn to the formalisation of all four clauses. Again, we must pay careful attention to the details. Our later verification demonstrates that the first two formalisations yield verified theorems, but we have left the matter of whether they are *too easily provable* to inspection. The formalisations are given in Figure 9.4.

Here, Theorems 9.3 and 9.4 formalise the fact that there are respectively at least two and at most two polygonal path-connected regions, while formulas 9.5 and 9.6 formalise the fact that exactly one region is unbounded.

We would like to draw the reader's attention to the side-condition in the conclusion of Theorem 9.3. We must assert that P and Q do not lie on the polygon. *Any* two points not satisfying this condition are such that they cannot be polygonal path-connected. Without the condition, the theorem is trivial.

$$\begin{aligned}
&\vdash \texttt{simple_polygon}\ \alpha\ Ps \\
&\quad \Longrightarrow \exists P.\ \exists Q.\ \texttt{on_plane}\ P\ \alpha \wedge \texttt{on_plane}\ Q\ \alpha \\
&\qquad \wedge \neg \texttt{on_polypath}\ Ps\ P \wedge \neg \texttt{on_polypath}\ Ps\ Q \\
&\qquad \wedge \neg \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ P\ Q
\end{aligned} \tag{9.3}$$

$$\begin{aligned}
&\vdash \texttt{simple_polygon}\ \alpha\ Ps \\
&\quad \wedge \texttt{on_plane}\ P\ \alpha \wedge \texttt{on_plane}\ Q\ \alpha \wedge \texttt{on_plane}\ R\ \alpha \\
&\quad \wedge \neg \texttt{on_polypath}\ Ps\ P \wedge \neg \texttt{on_polypath}\ Ps\ Q \wedge \neg \texttt{on_polypath}\ Ps\ R \\
&\quad \Longrightarrow \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ P\ Q \\
&\qquad \vee \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ P\ R \\
&\qquad \vee \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ Q\ R
\end{aligned} \tag{9.4}$$

$$\begin{aligned}
&\vdash (\forall P.\ \texttt{on_polypath}\ Ps\ P \Longrightarrow \texttt{on_plane}\ P\ \alpha) \\
&\Longrightarrow \exists a.\ \forall P.\ \texttt{on_line}\ P\ a \Longrightarrow \texttt{on_plane}\ P\ \alpha \wedge \neg \texttt{on_polypath}\ Ps\ P.
\end{aligned} \tag{9.5}$$

$$\begin{aligned}
&(\forall P.\ \texttt{on_line}\ P\ a \vee \texttt{on_line}\ P\ b \Longrightarrow \neg \texttt{on_polypath}\ Ps\ P \wedge \texttt{on_plane}\ P\ \alpha) \\
&\quad \wedge \texttt{on_plane}\ A\ \alpha \wedge \texttt{on_plane}\ B\ \alpha \\
&\quad \wedge \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ b \\
&\quad \Longrightarrow \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ A\ B.
\end{aligned} \tag{9.6}$$

Figure 9.4: The Polygonal Jordan Curve Theorem formalised

9.4 Conclusion

In this brief chapter, we have presented the verification goals for the following two chapters, namely Theorems 9.3 and 9.4. For the formalisations of these two theorems, we have chosen to identify polygons and polygonal paths with their vertex lists. This means that the definitions of simple polygons and the formalisations of the two theorems rely on a number of functions and theorems about lists.

The full formalisation can be presented in just a few pages of higher-order logic, and should be studied carefully to ensure that our verifications correspond to a proof of the Polygonal Jordan Curve Theorem. Otherwise, the suite of formal definitions is self-contained. Any new definitions and theorems we introduce from here on are just the scaffolding needed to support the verification.

Chapter 10

Verifying the Polygonal JCT: Part I

We now come to the first of our two main contributions. We must verify Theorem 9.3 from Hilbert's axioms of ordered geometry. We assume a simple polygon, and must find two points in the plane with the following property: given an arbitrary polygonal path connecting the two points, we can find another point at which the path intersects the simple polygon. The overall idea of this proof is very similar to Veblen's 1904 proof [100] which we described in detail in §8.4. We give the correct version of this proof now.

10.1 Sketch Proof

Consider the polygon Ps shown in Figure 10.1. We pick an arbitrary point O between P_1 and P_2 and then cast an arbitrary ray h from O to a segment of the polygon other than P_1P_2. Of all the intersections that h makes with the polygon, we pick the one closest to O and label it H. We then pick an arbitrary point A between O and H. For this point, we have that the segment AO does not intersect the polygon Ps. Finally, we consider the ray emanating from O in the other direction. Applying the same reasoning as above, we find a point B such that BO does not intersect Ps. We end up with a segment AB which intersects the polygon Ps exactly once between P_1 and P_2, namely at the point O.

Now consider any polygonal path which connects A and B. Together with the segment AB, this yields another polygon Qs (possibly non-simple) that intersects Ps at least once at the segment P_1P_2. We now proceed by considering the exact same sequence of triangles that appear in Veblen's proof. However, the observation we shall carry through the argument is that the closed polygon Qs must cross the edges of any triangle

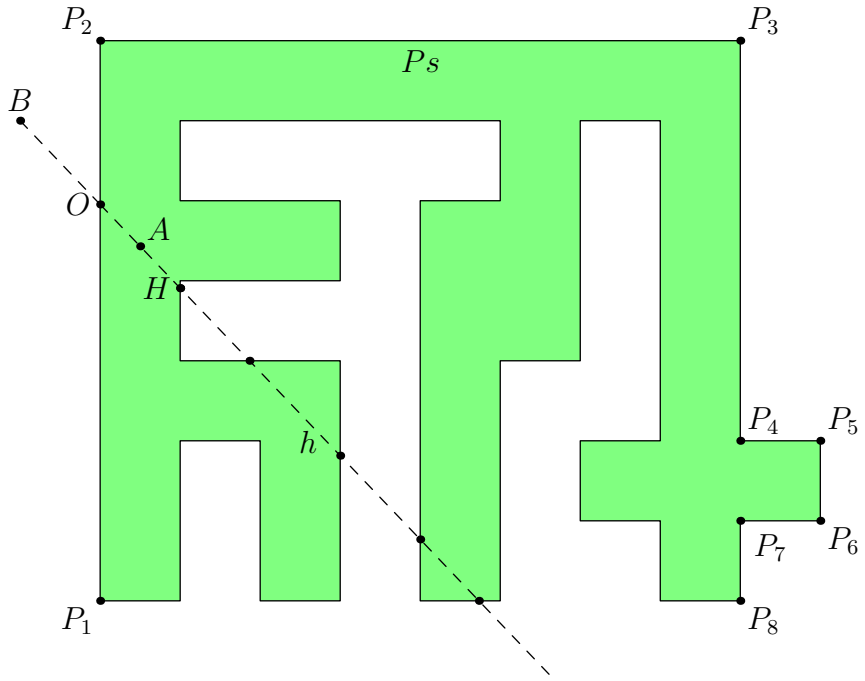


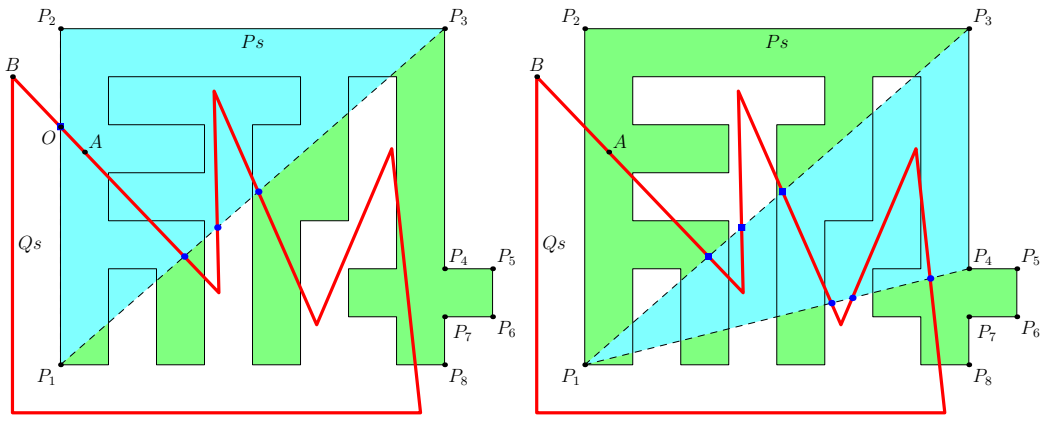
Figure 10.1: The witnesses (A and B) for Theorem 9.3

an even number of times. This should be intuitively obvious. Indeed, every time the edge of a polygon crosses an edge of a triangle, it changes from being inside to outside the triangle and *vice versa*. The total crossings must therefore be even in number, since we end in the same region we started.

In Figures 10.2 and 10.3, we illustrate a run of this parity argument through the triangles $P_1P_2P_3$, $P_1P_3P_4$, $P_1P_5P_6$, $P_1P_7P_8$ (the steps for the triangles $P_1P_4P_5$, $P_1P_6P_7$ have been omitted for clarity). At the start of the proof, we assume that the polygon Qs crosses the edge P_1P_2 exactly once and at the point O. Note however, that for the purposes of the argument, we only need the more general fact that it crosses an odd number of times.

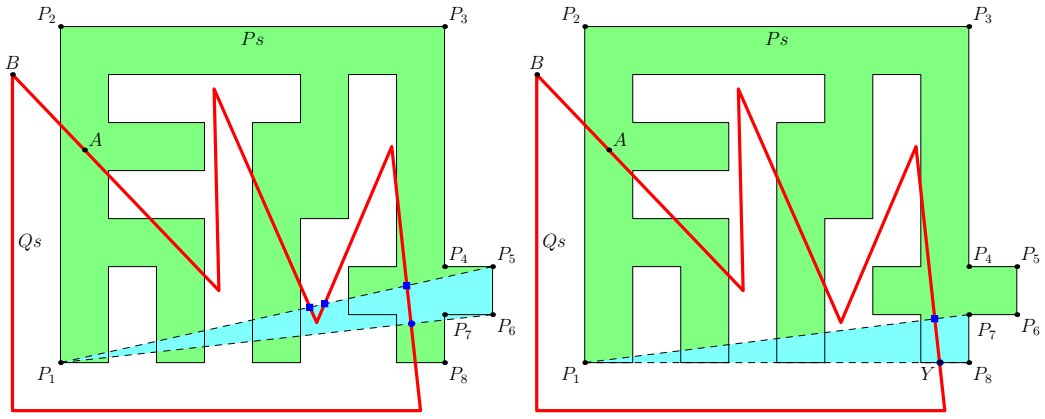
Now, if Qs intersects the edge P_2P_3, we are done. Thus, we can assume that it does not cross this edge. In that case, the polygon Qs must cross the edge P_1P_3, also an odd number of times, to ensure that the total crossings are even. Indeed, there are 3 such crossings shown in Figure 10.2(a). Hence, we can continue with the triangle $P_1P_3P_4$.

We then have that the polygon Qs must cross the triangle $P_1P_3P_4$ an even number of times. We know that it crosses P_1P_3 an odd number of times, and we can assume that it does not cross P_3P_4 (otherwise, we are done). Hence, it must cross P_1P_4 an odd number of times, in order that the total be even. Indeed, there are another three crossings shown



(a) Step 1: Four crossings (b) Step 2: Six crossings

Figure 10.2: Parity proof



(a) Step 4: Four crossings (b) Step 6: Two crossings

Figure 10.3: Parity proof (continued)

in Figure 10.2(b). Again, we continue with the next triangle. Eventually, we shall find a point of intersection with the polygon, shown as point Y in Figure 10.3.

This is a deceptively simple proof. Most of the work involved hinges on the informal notion of "crossing". In the next section, we shall show how this notion is formulated.

10.2 Formulation: Crossings

We hope that our use of "crossing" in the above is intuitively clear. The basic idea is that a polygonal path crosses a segment AB when it intersects AB and moves from one side to the other. While intuitive, we found the idea resisted a nice formulation.

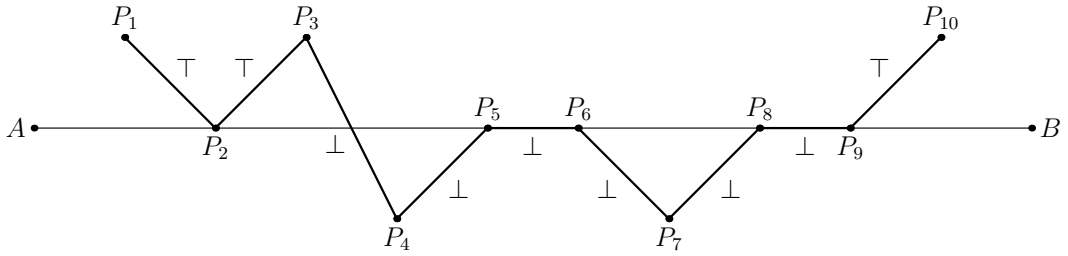


Figure 10.4: Assignment of context on a segment

10.2.1 Context

In our formulation, a polygonal path is a vertex list, and from this vertex list it is trivial to recover an edge list using the function `adjacent`. Our plan then is to use this edge list to compute the number of times the path crosses the segment AB by reducing it to the problem of computing the number of times a single edge of the path crosses AB. We can then define the crossings of the full polygonal path by summing the crossings at each of its edges.

There is an annoying problem. Suppose we have an edge P_iP_{i+1} of a polygonal path, and a triangle ABC, and suppose we are interested in whether P_iP_{i+1} crosses the triangle at AB. If there is a point on AB which is strictly between P_i and P_{i+1}, then we know there is a crossing at P_iP_{i+1}. But if one of the endpoints P_i or P_{i+1} are points on AB, then it is not the segment P_iP_{i+1} which crosses the triangle, but a potentially larger polygonal path $P_{i-m}\dots P_{i-1}P_iP_{i+1}\dots...P_{i+p}$ with $m \geq 0, p > 0$.

We can preserve the idea that the presence or absence of a crossing is nevertheless defined for each edge of a polygonal path by introducing a *context* variable Γ : `bool`, and assign a value of this variable to each edge P_iP_{i+1} of the polygonal path. The value will tell us on which side of AB the endpoint P_{i+1} lies. In the peculiar case that P_{i+1} lies on AB itself, we can just propagate the preceding context.

In Figure 10.4, we show a context value assigned in this way to the edges of $P_1 \dots P_{10}$. A value of $\top$ indicates the side which is the top half of the diagram, while $\perp$ indicates the bottom half. There are two places where the context switches truth value, which indicates that the polygonal path crosses the segment AB twice.

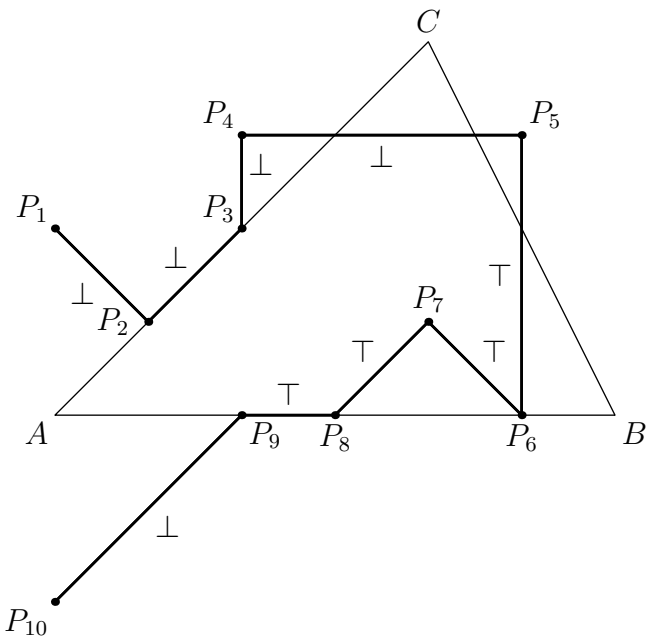


Figure 10.5: Assignment of context in a triangle

10.2.2 Combined Context for Triangles

We will be counting crossings on the edges of a triangle, which will require three context variables, one for each edge. Also, the assignment of $\top$ and $\bot$ to the sides of each edge of the triangle cannot be arbitrary as it was in Figure 10.4. The problem with the triangle case is that we want to reason about the total crossings on all three sides and thus consider the way the variables interplay.

For each edge, we shall therefore declare the $\top$ side for the corresponding context to be the side (or half-plane) containing the triangle's interior. We will then simply combine the contexts by taking their conjunction. One way to think about this is that we are using a single context variable which tracks whether a segment ends inside or outside the triangle.

In Figure 10.5, we show the assignment of the combined context value to each vertex of a polygon intersecting a triangle ABC. Here, if an edge P_iP_{i+1} is such that P_{i+1} lies outside the triangle, then the edge is assigned $\bot$. If P_{i+1} lies inside the triangle, then the edge is assigned $\top$. The only complication is how to set the context when the point P_{i+1} lands on an edge. For this, we think about the three component contexts for each edge of the triangle in terms of the ideas from the previous subsection.

Take the segment P_5P_6. The point P_6 lands on the edge AB, so the component context for this edge is propagated from the previous segment. In particular, since P_5 lies on

the side of AB which contains the triangle's interior, we propagate the value $\top$.

The component of the context for the side BC is simply $\top$, since P_6 is on the side of BC containing the triangle's interior. The component of the context for AC is also $\top$, since P_6 lies on the side of AC containing the triangle's interior. Since the three component contexts for P_5P_6 are all $\top$, so is the combined context.

To bring these ideas together, we will compute the number of crossings at each edge of the triangle as follows: first, we count a crossing for P_iP_{i+1} and the edge AB (or AC or BC) every time there is a point on AB which is strictly between P_i and P_{i+1}. The only other crossings occur when P_i lies on the segment AB. Here, we count a crossing in two circumstances:

- the context was last $\bot$ and P_iP_{i+1} has a point in the interior of $\triangle ABC$ (and thus moves from outside to inside);
- the context was last $\top$ and P_iP_{i+1} has a point in the exterior of $\triangle ABC$ (and thus moves from inside to outside).

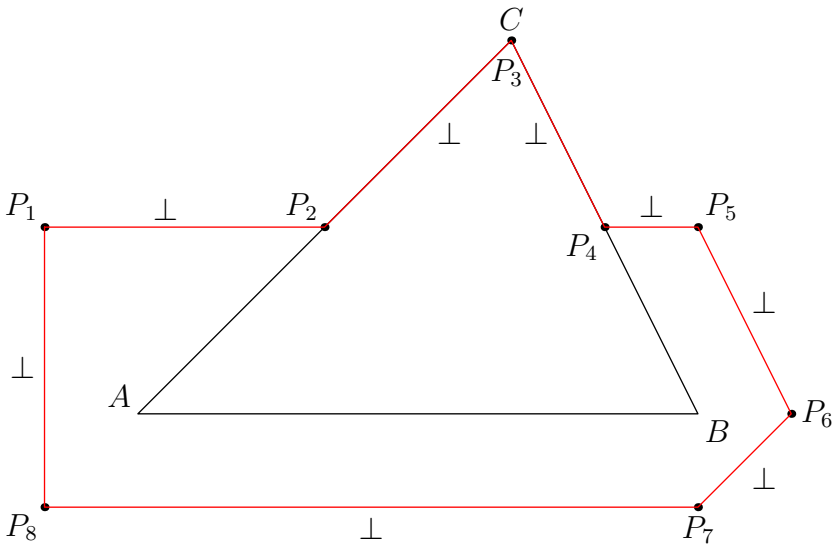
Thus, in Figure 10.5, there is one crossing on AC, two crossings on BC, and one crossing on AB.

Now that we are always counting crossings at an edge relative to a triangle, it might appear that we have rendered our notion too specific. We will still need to be able to count crossings at an arbitrary segment AB without mentioning triangles. To facilitate this, we show in §10.5.3 that once we have fixed the vertices AB in a triangle, our count of crossings at AB is independent of the choice of the vertex C. In other words, the expression "crossings of P_iP_{i+1} at the edge AB" is still well-defined, subject to a constraint on vertices detailed below. This fact, together with other key theorems (see Figure 10.25 later), should fully clarify the intended semantics of a "crossing."

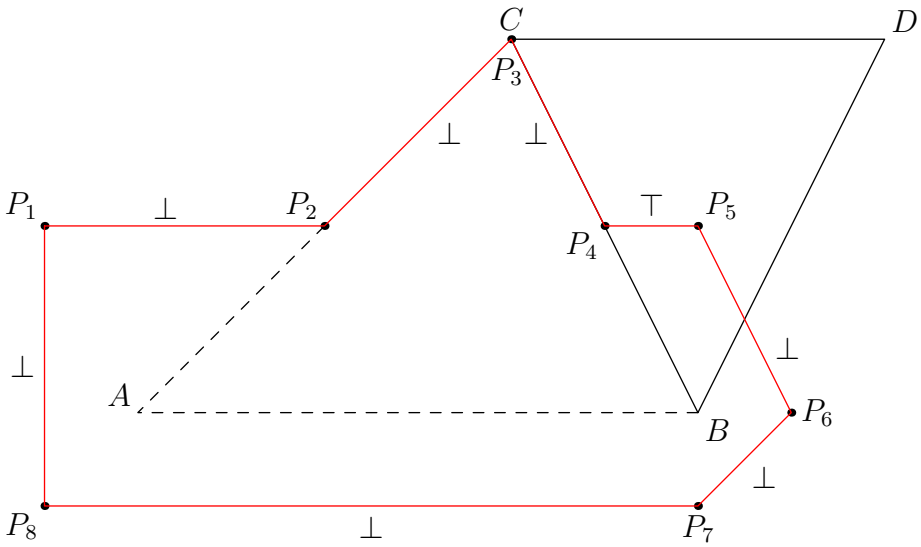
10.2.3 Avoiding Vertices

If we want the count of crossings at a particular edge of a triangle to be invariant of the position of the third vertex, we have a problem. Consider the scenario in Figure 10.6(a). Here, we have drawn a polygon $P_1P_2P_3P_4P_5P_6P_7P_8$ intersecting a triangle ABC. We have assigned our context appropriately to each segment, and concluded, quite reasonably, that the polygon does not cross ABC.

However, when we assign context values for the triangle BCD in Figure 10.6(b), we



(a)



(b)

Figure 10.6: Context with vertex crossings

find that there suddenly appears a crossing on the shared edge BC at the point P_4. In other words, the number of crossings at the segment BC is not well-defined on our scheme.

This difficulty can be eliminated quite simply by assuming that the polygonal path does not intersect any vertex of the triangle. We can get away with this because the vertices of the triangles we consider in our sketch proof are all vertices of the original polygon Ps. If at any time we found a point of intersection between the polygonal path and a vertex of one of the triangles, we will have found the desired point of intersection between the polygonal path and Ps.

10.2.4 Formalisation

We now introduce the functions with which we shall calculate the number of crossings against an edge of a triangle. They are supplied for the curious reader to eliminate any potential ambiguity in our explanation of contexts, and to give an indication of the distance between the intuitive and the formal idea of a crossing.

Our first function computes the crossings at an edge of a triangle based on a context.

$$\vdash_{def} \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}$$

$$= \begin{cases} 0, & \text{if } \texttt{between}\ A\ P_i\ B \wedge \texttt{between}\ A\ P_{i+1}\ B \\ 1, & \text{else if } \exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{between}\ A\ R\ B \\ 1, & \text{else if } \texttt{between}\ A\ P_i\ B \\ & \quad \wedge (\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \\ & \qquad \wedge\ \texttt{in\_triangle}\ (A,B,C)\ R \iff \neg\Gamma) \\ 0, & \text{otherwise.} \end{cases} \tag{10.1}$$

The first argument gives the three points defining the triangle we are interested in as a triple. We arbitrarily declare the first two components of this triple to be the edge of the triangle against which we want to compute crossings. The next argument is the context value Γ. The final two arguments are the endpoints of the polygonal path's edge against which we compute crossings.

Thus, to express the number of crossings for the edges AC and BC, we just use the terms $\texttt{crossing}\ (A,C,B)$ and $\texttt{crossing}\ (B,C,A)$ respectively, and to express the total

crossings of the segment P_iP_{i+1} on the triangle, we use the term

$$\texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1}.$$

Our next function computes the context value for a segment P_iP_{i+1} based on the last context. The arguments are the same, but here, the output does not depend on any particular ordering of the triple (A,B,C).

$$\begin{aligned}
&\vdash_{def}\ \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}\\
&\quad\Longleftrightarrow \texttt{in_triangle}\ (A,B,C)\ P_{i+1}\\
&\qquad\vee \left(\begin{array}{l}\texttt{on_triangle}\ (A,B,C)\ P_{i+1}\\ \quad\wedge \left(\begin{array}{l}(\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in_triangle}\ (A,B,C)\ R)\\ \quad\vee \texttt{on_triangle}\ (A,B,C)\ P_i \wedge \Gamma)\end{array}\right)\end{array}\right).
\end{aligned}$$

Finally, we define the function which will compute the total number of crossings of an arbitrary polygonal path against the edge AB for the triangle ABC. We do this recursively over the list of edges of the polygonal path, summing the values of `polypath_crossing` (A,B,C) for each segment and updating the context. Note that this function still requires an initial context Γ. We show where to get it from in §10.5.4.

$$\begin{aligned}
&\vdash_{def} \texttt{polypath_crossings}\ (A,B,C)\ \Gamma\ [] = 0\\
&\vdash_{def} \texttt{polypath_crossings}\ (A,B,C)\ \Gamma\ ((P_i,P_{i+1}) :: \mathit{segments})\\
&\qquad = \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}\\
&\qquad\quad + \texttt{polypath_crossings}\ (A,B,C)\ (\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1})\ \mathit{segments}.
\end{aligned}$$

10.3 Triangle Interiors

The above formulations and formalisation assume that we know how to express the interior, exterior and boundary of a triangle (respectively `in_triangle`, `on_triangle` and `out_triangle`). This we can do directly. Veblen, for instance, in his 1903 thesis [100], defined the interior of the triangle ABC as the set of points P such that there is a point X on the segment AB and a point Y on AC with P between X and Y (see Figure 10.7). Here is another definition: the interior of $\triangle ABC$ is the set of all points on the same side of AB as C, on the same side of AC as B and on the same side of BC as A. In other words, the interior of a triangle is the intersection of three half-planes.

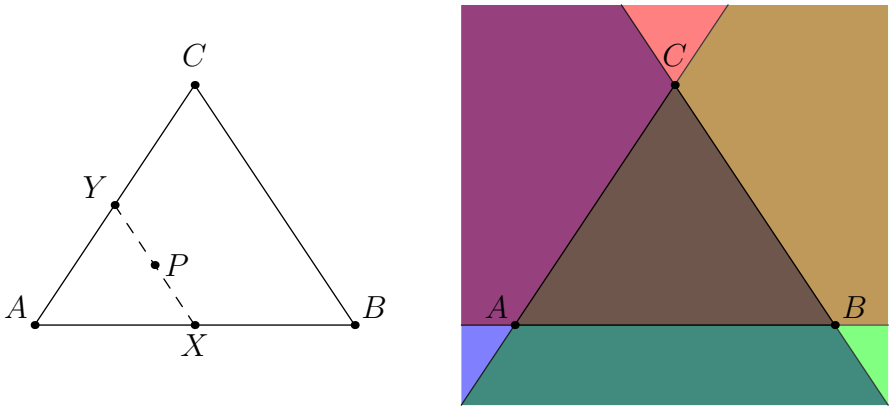


Figure 10.7: Two definitions of a triangle's interior

The two formulations can be formalised as:

$$\texttt{in\_triangle\_veblen}\ (A,B,C)\ P \iff$$
$$\exists X.\ \exists Y.\ \texttt{between}\ A\ X\ B \wedge \texttt{between}\ A\ Y\ C \wedge \texttt{between}\ X\ P\ Y.$$

$$\vdash_{def} \texttt{in\_triangle}\ (A,B,C)\ P \iff$$
$$\exists hp.\ \exists hq.\ \exists hr.\ \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hp)$$
$$\wedge\ \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hp)$$
$$\wedge\ \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hq)$$
$$\wedge\ \texttt{on\_line}\ C\ (\texttt{line\_of\_half\_plane}\ hq)$$
$$\wedge\ \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hr)$$
$$\wedge\ \texttt{on\_line}\ C\ (\texttt{line\_of\_half\_plane}\ hr)$$
$$\wedge\ \texttt{on\_half\_plane}\ hp\ C \wedge \texttt{on\_half\_plane}\ hq\ B \wedge \texttt{on\_half\_plane}\ hr\ A$$
$$\wedge\ \texttt{on\_half\_plane}\ hp\ P \wedge \texttt{on\_half\_plane}\ hq\ P \wedge \texttt{on\_half\_plane}\ hr\ P. \tag{10.2}$$

Veblen's definition is significantly shorter, but we wanted to try leveraging our theory of half-planes as much as possible in our verification of the Polygonal Jordan Curve Theorem, and the second definition gives us direct information about these. Besides which, the existentials in Veblen's definition do not have unique witnesses, while ours do, making Veblen's definition more complicated to reason with. Consider that it is not immediately clear that his definition is symmetric up to permutations of A, B and C. To prove this, we would need to figure out how to move from the arbitrary X and Y on AB and AC satisfying the given condition (and there are infinitely many

possible choices), to another X' and Y' on another choice of segments. With the second definition, the symmetry is almost immediate. In fact, HOL Light's `MESON` can easily prove the rewrites needed to normalise expressions of the form `in_triangle` (A,B,C).

$$\begin{aligned} \vdash \texttt{in\_triangle}\ (A,B,C)\ P &\iff \texttt{on\_triangle}\ (A,C,B)\ P \\ \wedge\ \texttt{in\_triangle}\ (A,B,C)\ P &\iff \texttt{on\_triangle}\ (B,A,C)\ P. \end{aligned} \tag{10.3}$$

That said, with our definition, our early verifications about triangles always became bloated in the same clumsy way. When we started from a hypothesis that a point lies inside a triangle, we found ourselves having to extract all three witnessed half-planes in the definition and all twelve conjuncts they satisfy. In many cases, we found that the the main body of the proof was shorter than the bloated statement of the assumptions. It might be suggested that this ugliness could have been avoided had we persevered instead with Veblen's definition, and tried to avoid reasoning about half-planes. We have some circumstantial against this: if Veblen's definition were a more useful starting point for reasoning about the interiors of triangles, we would expect that his formulation would appear as an immediate step in our proofs, whereupon we could reason more effectively about interiors. But this was never the case. There was no need obtain the two point witnesses given in Veblen's definition. Conversely, there were twelve places in our verifications where we had two points that could satisfy Veblen's definition, and where we appealed to a lemma which shows Veblen's definition implies our own. This suggests that our formulation is the more useful starting point.

We make one final remark about our definition, which is important to keep in mind for some of the later verification. Whenever we have `in_triangle` $(A,B,C)\ P$, we know that all triples chosen from $\{A,B,C,P\}$ are non-collinear. This means that explicit assumptions about non-collinearity can be suppressed in many of our verified theorems. It also means we can implement a discoverer `add_in_triangle` to derive these non-collinear triples automatically and make them available to the `obviously` primitive.

We end this subsection by considering the formalisation of a triangle's boundary and exterior. Since the boundary is just a polygonal path, it suffices to define:

$$\vdash_{def} \texttt{on\_triangle}\ (A,B,C)\ P \iff \texttt{on\_polypath}\ [A,B,C,A]\ P. \tag{10.4}$$

In fact, it is useful to reuse `on_polypath` in this way in other places. For instance, we can refer to the set of points of a segment AB with `on_polypath` $[A,B]$, and given

a triangle ABC, we can write `on_polypath` $[A,B,C]$ to refer to the points on just two sides of the boundary. Such formulas will prove convenient later on in our verification.

Finally, the exterior of a triangle can be defined simply as the set of points not on the triangle and not on the boundary. This definition classifies all points which are not on the plane as part of the exterior, but since we relativise all of our theorems against a single plane, this does not matter.

$$\vdash_{def} \texttt{out\_triangle}\ (A,B,C)\ P \iff \neg\texttt{in\_triangle}\ (A,B,C)\ P \wedge \neg\texttt{on\_triangle}\ (A,B,C)\ P.$$

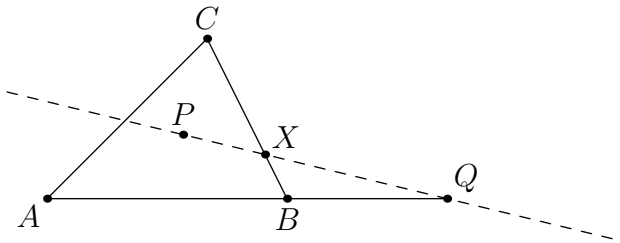
10.4 Some Preliminary Theorems

We recall the basic approach to synthetic axiomatic geometry as divided into two kinds of reasoning step: ones which introduce geometrical entities and ones which identify salient properties of the resulting figures. These properties allow us to introduce new geometrical entities, establish facts about them from which we can introduce new entities, and so on, until we have verified our goal theorem.

We have six theorems for triangle interiors, two to introduce points and four to reason about such points with respect to triangle interiors. In this section, we shall look in detail at a verification of one of the introduction theorems, and then summarise the remaining ones.

10.4.1 The Base Case

Our main goal in this chapter is to show that a simple polygon divides the plane into at least two regions. In the simplest case, we take the polygon to be a *triangle*, and we find that a triangle divides the plane much as a line divides the plane into half-planes. Specifically, given two points in different half-planes, we know there is a point between them which lies on the boundary, meaning we have an introduction theorem. There is an analogous introduction theorem for triangles, which we use frequently. It is even needed to prove our other introduction theorem (10.7) in §10.4.4. In turn, this second introduction theorem is crucial to the verification of the well-definedness of crossings



$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ P \wedge \texttt{between}\ A\ B\ Q\\ &\quad\Longrightarrow \exists X.\ \texttt{between}\ P\ X\ Q \wedge \texttt{between}\ B\ X\ C\end{aligned} \tag{10.6}$$

Figure 10.8: "Inner Pasch" for an interior point

at a triangle's side (see §10.5.3):

$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ P \wedge \texttt{out\_triangle}\ (A,B,C)\ Q\\ &\qquad\Longrightarrow \exists R.\ \texttt{on\_triangle}\ (A,B,C)\ R \wedge \texttt{between}\ P\ R\ Q.\end{aligned} \tag{10.5}$$

We initially hoped the proof of Theorem 10.5 would be trivial. After all, an almost identical theorem holds for half-planes, and a triangle is defined as the intersection of three of these. Instead, we found ourselves needing a point introduction lemma.

10.4.2 An "Inner Pasch" Lemma

Initially, our only means to introduce points relative to triangles was by Pasch's axiom (II, 4). But this axiom is often difficult to apply because it has a disjunctive conclusion. Luckily, there are easier versions to apply, namely the inner and outer variations, which we have derived as Theorems 5.2 and 5.3. Our point introduction lemma can be thought of as a variation of Theorem 5.3. It says that, given an interior point P of a triangle ABC, and a point Q outside the triangle on the ray AB, we can introduce the point X at which the line PQ intersects BC (we could then use the Outer Pasch Axiom to find the point at which PQ intersects AC). See Figure 10.8.

The verification of this lemma illustrates some common patterns of reasoning with half-planes, and some of the pros and cons of our representation. The first half of the verification is shown in Figure 10.9, where we obtain the three half planes defining the triangle. We must explicate the verbose constraints on these half-planes, before showing that the lines of each lie in the plane α. These facts are needed in order to infer the defining property of each half-plane, namely that two points in the plane α are

in the same half-plane precisely when their segment does not cross the line of the half-plane. The annoyance here is that we really do not care about such details, since all our assumptions should constrain the figure to the plane α anyway. If we transcribed these proofs to planar geometry, these details could be omitted, but for now, they show up as a weakness in our representation.

`theorem` on_plane $A\ \alpha \wedge$ on_plane $B\ \alpha \wedge$ on_plane $C\ \alpha$	
in_triangle $(A,B,C)\ P \wedge$ between $A\ B\ Q$	
$\Longrightarrow \exists X.$ between $P\ X\ Q \wedge$ between $B\ X\ C$	
`assume` $\neg(\exists a.$ on_line $A\ a \wedge$ on_line $B\ a \wedge$ on_line $C\ a)A$ `by` (C.1)	0
`assume` on_plane $A\ \alpha \wedge$ on_plane $B\ \alpha \wedge$ on_plane $C\ \alpha$	1,2,3
`assume` in_triangle $(A,B,C)\ P$	
`so consider` hp, hq and hr `such that`	
on_line A (line_of_half_plane hp) $\wedge$ on_line B (line_of_half_plane hp)	4,5
on_half_plane $C\ hp \ldots \wedge$ on_half_plane $P\ hr$	6,7
… `by` (10.2)	8..15
`assume` between $A\ B\ Q$	16
`obviously by_neqs have` $\forall X.$ on_half_plane $hp\ X \Longrightarrow$ on_plane $X\ \alpha$	
`from` 0,1,2,3,4,5,6 `by` (I, 6),(7.5)	17
…	18,19

Figure 10.9: Proof of "Inner Pasch" for an interior point (part 1)

The rest of the proof is shown in Figure 10.10. In contrast to the first part of the proof, the steps here are succinct, readable and geometrically interesting. With the necessary assumptions laid out, we see how easily the theory of half-planes has been leveraged via Theorems 7.8 and 7.9. In contrast to the first part of the proof, this puts the use of half-planes in a much more positive light.

$$\begin{aligned} &\vdash \texttt{on\_line}\ P\ (\texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{on\_half\_plane}\ hp\ Q \\ &\quad \Longrightarrow \texttt{between}\ P\ Q\ R \vee \texttt{between}\ P\ R\ Q \Longrightarrow \texttt{on\_half\_plane}\ hp \end{aligned} \tag{7.8}$$

$$\begin{aligned} &\vdash \texttt{on\_half\_plane}\ hp\ P \wedge \texttt{on\_half\_plane}\ hp\ R \\ &\quad \Longrightarrow \texttt{between}\ P\ Q\ R \Longrightarrow \texttt{on\_half\_plane}\ hp\ Q \end{aligned} \tag{7.9}$$

```
obviously by_ncols hence on_plane Q α                                          20
    ¬on_line Q (line_of_half_plane hr)                                         21
    ¬on_half_plane hr Q from 0,1,2,12,13,14,16 by (I, 6),(II, 1),(7.7)         22
consider X such that on_line X (line_of_half_plane hr) ∧ between P X Q
 from 15,19,20,21,22 by (7.7)                                                  23
have on_line Q (line_of_half_plane hp) by (I, 2),(II, 1) from 4,5,16
hence on_half_plane hp X by (II, 1),(7.8) from 7,23                            24
hence ¬between C B X from 5,6,17 by (7.7)                                      25
hence on_half_plane hq Q from 8,10,16 by (7.8)
hence on_half_plane hq X from 11,23 by (7.9)
hence ¬between B C X ∧ B ≠ X ∧ C ≠ X from 5,9,10,18,24 by (7.7),(7.6)
obviously by_neqs qed from 0,12,13,23,25 by (THEOREM 4)
```

Figure 10.10: Proof of "Inner Pasch" for an interior point (part 2)

The basic strategy of the verification is to note that because A and Q lie on opposite sides of BC, so too must P and Q. Thus, we can find a point X between P and Q which is on the line BC. We just need to show that this point X lies more specifically between B and C.

To do this, we note that P and X lie on a ray emerging from the point Q on the line AB, and so they must be on the same side of this line. Thus P, C and X must all lie on the same side of AB which means that the point B cannot possibly lie between any of them. Similar considerations apply if we look at the line AC. We can thus conclude that X can only lie between B and C.

The verification captures the *structure* of this line of argument almost exactly. However, the terms are almost completely different. To begin with, we do not introduce anonymous rays. This would only add extra `consider` steps, which is unnecessary when we can talk directly in terms of betweenness. We also avoid talking in terms of sides of a line by talking instead in terms of half-planes. We effectively have the translations given in Figure 10.11.

With these translations in mind, we hope the reader is convinced that the informal argument and a model synthetic proof can be recovered systematically from the verifi-

`on_half_plane` *hp*	$\Longleftrightarrow$	on the same side of *AB* as *C* and *P*;
`on_half_plane` *hq*	$\Longleftrightarrow$	on the same side of *AC* as *B* and *P*;
`on_half_plane` *hr*	$\Longleftrightarrow$	on the same side of *BC* as *A* and *P*;
`line_of_half_plane` *hp*	$\Longleftrightarrow$	the line *AB*;
`line_of_half_plane` *hq*	$\Longleftrightarrow$	the line *AC*;
`line_of_half_plane` *hr*	$\Longleftrightarrow$	the line *BC*.

Figure 10.11: Interpretation of half-planes

cation. We can try to excuse the translation by drawing an analogy between synthetic proofs and their accompanying diagrams. The diagram is strictly unnecessary, but can easily be recovered by carefully following the prose, and it is often helpful to reconstruct it. Similarly, our informal argument can be easily recovered from our formal verification, substituting intuitive phrases such as "a ray emerging from the line" so that it is easier to follow, even if such phrases do not point to interesting abstractions that would help the theorem prover.

10.4.3 From "Inner Pasch" to the Base Case

We can now give an informal proof of Theorem 10.5. In Figure 10.12, we suppose that P is inside a triangle ABC and Q is outside. Then P is on the same side of one of the triangle's edges and opposite vertex, while Q is not. Let us suppose, without loss of generality, that P is on the same side of AB as C while Q is not. Then PQ must intersect the line AB. If PQ intersects the *segment* AB, we have found the required point on the triangle's boundary. Otherwise, there is a point R on the segment PQ which also lies on either the ray emanating from B in the direction $\overrightarrow{AB}$ or on the ray emanating from A in the direction $\overrightarrow{BA}$. By applying (10.6) to each case, we can then find a point on the side BC or the side AC respectively, and we are done.

We have made a without-loss-of-generality assumption in this argument, namely in our choice of AB and the point C. As Harrison has shown [37], such assumptions can often be handled elegantly using without-loss-of-generality tactics, particularly in geometry. However, these tactics typically exploit a *Kleinian View* of geometry. This view of geometry can be described as "subtractive" [2]: we start from a rich mathematical structure such as $\mathbb{R}^n$, and then ignore details by working only with invariants under a transformation group. Axiomatic geometry, on the other add, is additive, starting with

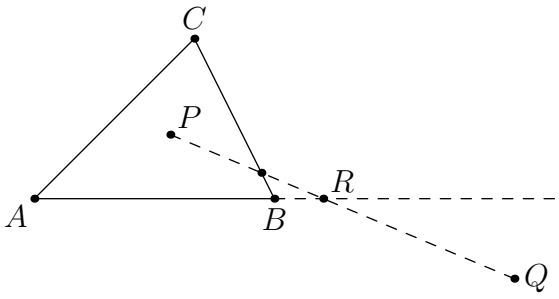


Figure 10.12: "Inner Pasch" to the base case

only the most primitive machinery. As such, it is not clear how to build a theory of invariants which could capture our without-loss-of-generality cases.

Instead, we formalised the above argument as a lemma, and then wrote an *ad hoc* procedural script to manually apply the symmetries. In ordered geometry, we only need to consider six symmetries and so the procedural boilerplate is hardly a bottleneck compared to our use of MESON in declarative proofs, but this is still somewhat inelegant compared to doing proper without-loss-of-generality reasoning.

10.4.4 Additional Theorems

We have one more theorem to introduce points. Here, we suppose that we have a point P on the edge AB of a triangle ABC and a point Q outside the triangle but on the same side of AB as C. In this case, the segment PQ must intersect the polygonal path $[A,B,C]$ at a point X (see Figure 10.13). The half-plane hp in this theorem is used to signify the side of AB on which the point C lies.

$$\begin{aligned}&\vdash \texttt{between}\ A\ P\ B\\ &\quad \wedge \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hp)\\ &\quad \wedge \texttt{on\_half\_plane}\ C\ hp \wedge \texttt{on\_half\_plane}\ Q\ hp\\ &\quad \wedge \texttt{out\_triangle}\ (A,B,C)\ Q \implies \exists X.\ \texttt{between}\ P\ X\ Q \wedge \texttt{on\_polypath}\ [A,B,C]\ X.\end{aligned} \tag{10.7}$$

The remaining four theorems assume we have a configuration of points in relation to a triangle, and conclude that one of the points is interior or exterior. These theorems are used routinely throughout the first half of our main verification, particularly when we come to counting how many times a polygonal path crosses the sides of a triangle (see

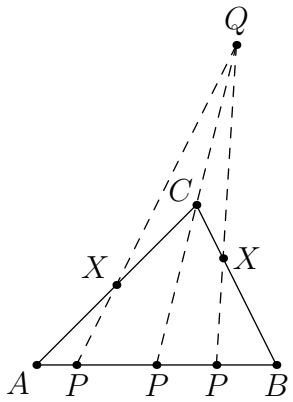
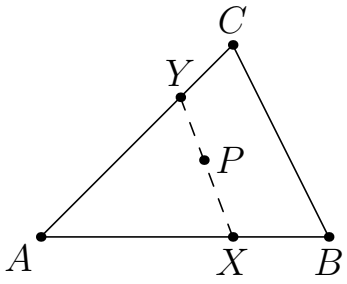
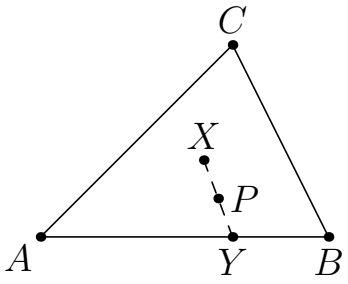


Figure 10.13: Another point introduction theorem



(a) Points between the sides of a triangle are interior (10.8)



(b) Points between an interior point and a side are interior (10.9)

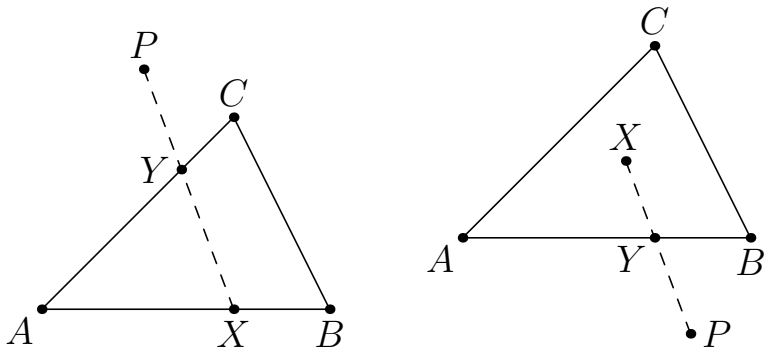
Figure 10.14: Triangle interior theorems

§10.5). Their proofs are similar to the one given in the previous section, and always reduce to reasoning about the interaction between rays and half planes.

We give diagrams and a short description for each theorem in Figures 10.14 and 10.15. These theorems all have reasonably clear synthetic verifications, and together require 82 verification steps. Roughly two fifths of these steps are assisted by our incidence discoverer via the `obviously` and `clearly` primitives.

Note that Theorem 10.8 is one direction of the equivalence between our definition of triangles and Veblen's (see §10.3).

$$\begin{aligned} \vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ \wedge \texttt{between}\ A\ X\ B \wedge (\texttt{between}\ A\ Y\ C \vee C = Y) \\ \Longrightarrow \texttt{between}\ X\ P\ Y \Longrightarrow \texttt{in\_triangle}\ (A,B,C)\ P. \end{aligned} \tag{10.8}$$



(a) A ray through an opposite side leaves the triangle (10.10)

(b) A ray from inside the triangle to a side emerges outside the triangle (10.11)

Figure 10.15: Triangle exterior theorems

$$\vdash \texttt{in\_triangle}\ (A,B,C)\ X \wedge \texttt{on\_triangle}\ (A,B,C)\ Y \\ \Longrightarrow \texttt{between}\ X\ P\ Y \Longrightarrow \texttt{in\_triangle}\ (A,B,C)\ P. \tag{10.9}$$

$$\vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ \wedge \texttt{between}\ A\ X\ B \wedge \texttt{between}\ A\ Y\ C \\ \Longrightarrow \texttt{between}\ X\ Y\ P \Longrightarrow \texttt{out\_triangle}\ (A,B,C)\ P. \tag{10.10}$$

$$\vdash \texttt{in\_triangle}\ (A,B,C) X \wedge \texttt{on\_triangle}\ (A,B,C)\ Y \\ \Longrightarrow \texttt{between}\ X\ Y\ P \Longrightarrow \texttt{out\_triangle}\ (A,B,C)\ P. \tag{10.11}$$

10.5 Key Theorems of Crossings

The formal definition of crossings as the threading of a context variable through a sequence of conditionals takes us a *long way* from the intuitive idea. The intuition only reappears in our key theorems governing the definition, and the distance between the intuition and the formalisation can be measured by the thousand or so lines of mostly declarative proof and the enormous number of case-splits we consider to bridge the gap.

10.5.1 Numbers of Crossings

First, a relatively simple matter: a single segment crosses a triangle at most twice. Our verification of this takes the form of a crisp declarative proof based on Bernays' supplement (7.3) that we discussed in §7.2.2. We do not need any messy case-splits, only a short piece of procedural script to eliminate without-loss-of-generality assumptions. We end up with this:

$$\begin{aligned}\vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\ \Longrightarrow \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1}\\ + \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} \leq 2.\end{aligned}$$

The only unpleasantness comes from unfolding the definition of `crossing`, which requires that we face the mess of case-splits from Definition 10.1. For this, we use a `tactics` step and a tactic `unfold_crossing_tac` which unfolds the definition of `crossing` and then sweeps through the goal term eliminating the cases. Again, this tactic does not modify any assumptions, and it is typically only applied at the very start of a verification. With the cases converted, the `assume` steps allow us to make more meaningful assumptions, as in the verification extract in Figure 10.16.

$$\begin{aligned}&\texttt{theorem}\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad \wedge \texttt{crossing}\ (A,B,C)\ X\ P_i\ P_{i+1} = 1\\ &\quad \wedge \texttt{crossing}\ (A,C,B)\ X\ P_i\ P_{i+1} = 1\\ &\quad \Longrightarrow \texttt{crossing}\ (B,C,A)\ X\ P_i\ P_{i+1} = 0\\ &\texttt{assume}\ \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\ &\texttt{tactics unfold\_crossing\_tac}\\ &\texttt{assume between}\ A\ P_i\ B \vee \exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{between}\ A\ R\ B\end{aligned}$$

Figure 10.16: Unfolding crossings

Things get *really* hairy for our next theorem, which clearly explains how the values of `crossing` compare when evaluated for a single segment at the various sides of a triangle. We give an impression of the cases involved in Figure 10.17.

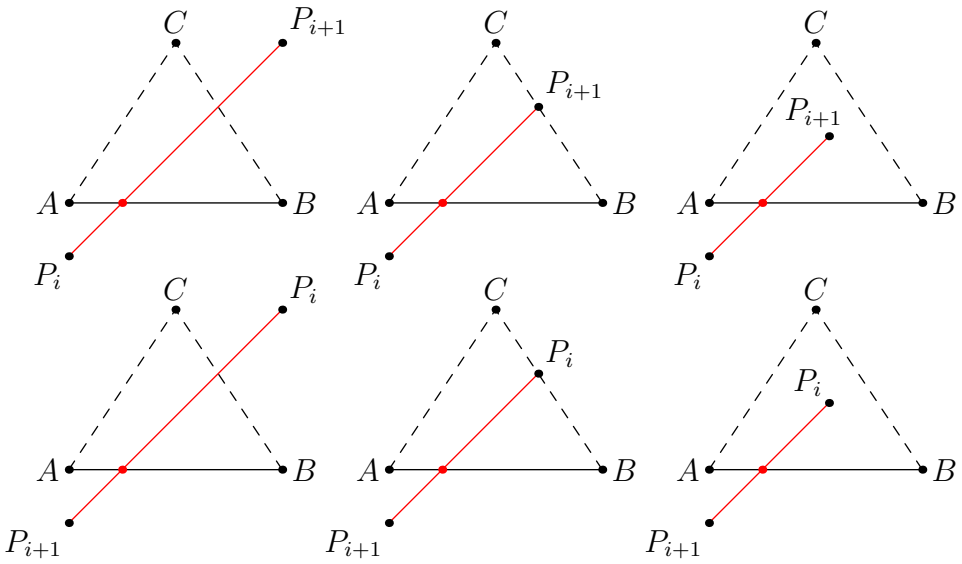


Figure 10.17: Cases of crossings

$$
\begin{aligned}
\vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)& \\
\wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ A \wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ B& \\
\wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ C& \\
\wedge (\neg\texttt{on\_triangle}\ (A,B,C)\ P_i \implies (\texttt{in\_triangle}\ (A,B,C)\ P_i \iff \Gamma))& \\
\implies \left(\begin{array}{l}
\texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1} \\
\qquad + \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} = 1 \\
\iff \Gamma = \neg\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}
\end{array}\right)&
\end{aligned}
$$

The first hypothesis just requires that ABC is a triangle. The second requires that the segment P_iP_{i+1} does not intersect any of the vertices, as per our discussion in §10.2.3. The rest of the theorem then clarifies both the idea behind a crossing and the idea behind the context variable Γ. The conclusion says that the sum of crossings at the three sides is 1 precisely when the context variable switches truth value. The formalisation almost transparently captures a claim made in the sketch proof: "every time the edge of a polygon crosses an edge of a triangle, it changes from being inside to outside the triangle and *vice versa*."

There is one more thing we should say about the context Γ. The theorem hypothesises that when P_i is not on the sides of a triangle then Γ tracks whether the point is inside or outside. Since P_i is intended to be a vertex of a polygonal path and P_iP_{i+1} an edge,

we want to make sure that this hypothesis on Γ is preserved as it threads through the remaining edges.

Because a vertex of the polygon P_{i+1} is the successor of P_i, what we are saying here is that, just as Γ tracks whether P_i is inside or outside the triangle, so too must Γ_{next} track whether P_{i+1} is inside or outside. This matter is settled trivially from the definition using the simplifier.

$$\vdash \neg\texttt{on\_triangle}\ (A,B,C)\ P_{i+1}$$
$$\Longrightarrow\ (\texttt{in\_triangle}\ (A,B,C)\ P_{i+1} \iff \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}).$$

10.5.2 Overview of Some Verification

Rather than go into all the details of the verification, we will give a typical extract of a specific case, showing how in these proofs we are still relying on our discovery algebra from Chapter 4 and our linear ordering tactic from Chapter 6. We also see how we leverage our lemmas for this section, and thus avoid having to deal directly with half-planes.

The case we consider is equivalent to saying that if a segment P_iP_{i+1} crosses a triangle ABC exactly once between P_i and P_{i+1} at AB, then one of P_i and P_{i+1} is interior to the triangle while the other is exterior.

$$\begin{aligned}
&\neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C\ a)\\
&\wedge \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{between}\ A\ R\ B\\
&\quad \wedge \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1} = 0 \wedge \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} = 0\\
&\wedge \neg\texttt{on_polypath}\ [P_i,P_{i+1}]\ A \wedge \neg\texttt{on_polypath}\ [P_i,P_{i+1}]\ B\\
&\quad \wedge \neg\texttt{on_polypath}\ [P_i,P_{i+1}]\ C\\
&\wedge \neg\texttt{on_triangle}\ (A,B,C)\ P_i \wedge \neg\texttt{on_triangle}\ (A,B,C)\ P_{i+1}\\
&\qquad \Longrightarrow\ (\texttt{in_triangle}\ (A,B,C)\ P_i \iff \texttt{out_triangle}\ (A,B,C)\ P_{i+1}).
\end{aligned} \tag{10.12}$$

We divide the verification into the three cases shown in Figure 10.18. In case (a), we have assumed that P_i is interior. It then follows immediately from Theorem 10.11 that P_{i+1} is exterior. In case (b), we have assumed that P_i is exterior and that P_i and P_{i+1} are in line with the vertex C. In this case, we just apply Theorem 10.8. Finally, in case (c), we have assumed that P_i is again exterior but that the line of P_iP_{i+1} does not intersect C. Under these circumstances, we can apply Pasch's Axiom (II, 4) to the triangle and

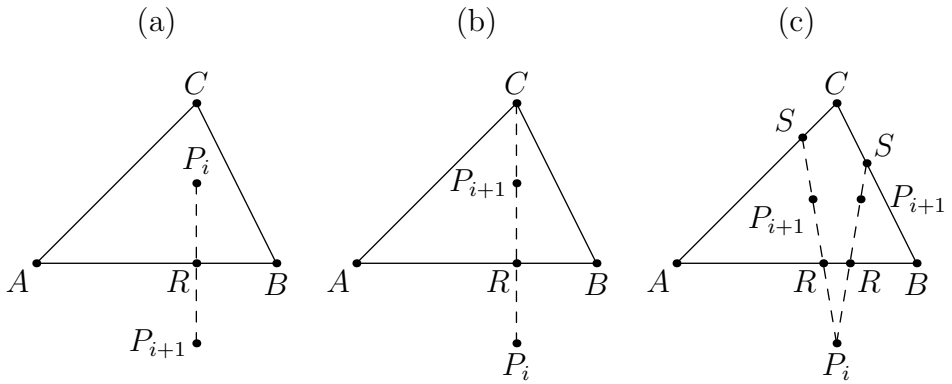


Figure 10.18: Main case-split

the line of P_iP_{i+1} using our discoverer `by_pasch` and thus obtain a point S either on AC or BC. It then follows from Theorem 10.8 that P_{i+1} is interior.

Actually, things are not *quite* so simple for cases (b) and (c). In order to apply Theorem 10.8 in case (b), we first have to prove that P_{i+1} is between C and R. To do this, we want to apply our linear ordering tactic, but for this to work, the tactic will need some facts about the existing order relations among the points P_i, P_{i+1}, R and C. These facts come from various places.

First off, the incidence discoverer tells us that $C \neq R$. Next, from (10.9) and the fact that P_i is exterior, we conclude that P_i does not lie between C and R. Finally, since P_iP_{i+1} does not intersect C, we know that all three points are distinct and that C does not lie between P_i and P_{i+1}. Each of these inferences corresponds to a single declarative step, and once in place, the linear reasoning tactic can be applied to the four points C, P_i, P_{i+1} and R, where it is able to show that P_{i+1} lies between C and R. We finish by applying (10.8) to show that P_{i+1} is interior to the triangle.

Case (c) is more involved, but the most interesting part is probably that which establishes that neither A nor B lie on the line of P_iP_{i+1}. Here, we proceed by contradiction, once for A and once for B. We can solve the goal in one step with the linear reasoning tactic, provided we again obtain the necessary information for it to do its work.

For instance, assuming that A, P_i and P_{i+1} lie on a line, the linear reasoning tactic will first infer that A, B, P_i, P_{i+1} and R are all collinear. If it knew further than P_i does not lie on the segment AB, it would conclude that one of A or B lies on the segment P_iP_{i+1}, which *we* know to be impossible. This suggests that we should seed the tactic with the following facts, with which it solves the goal by reasoning about the ordering of A, B,

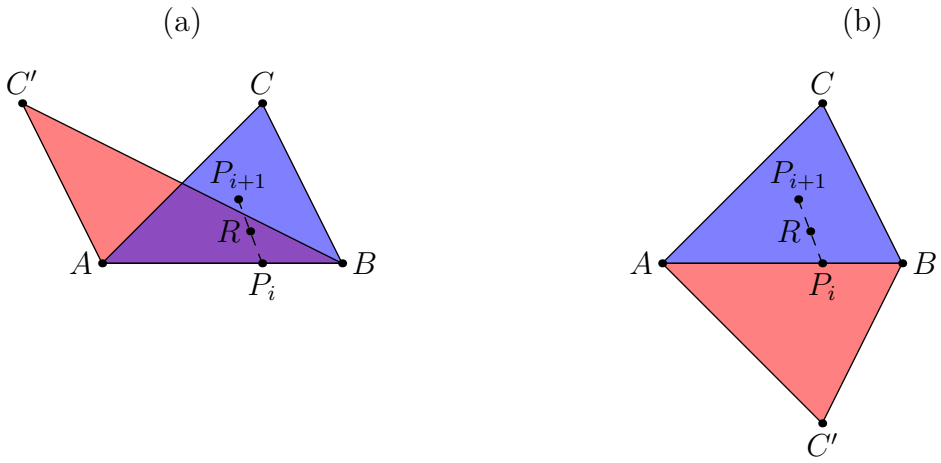


Figure 10.19: Triangles sharing an edge

P_i, P_{i+1} and R:

$$A \neq P_i \wedge A \neq P_{i+1} \wedge B \neq P_i \wedge B \neq P_{i+1} \\ \wedge \neg\texttt{between}\ P_i\ A\ P_{i+1} \wedge \neg\texttt{between}\ P_i\ B\ P_{i+1} \wedge \neg\texttt{between}\ A\ P_i\ B.$$

10.5.3 Crossings are Well-defined

In our sketch proof from §10.1, we implicitly assume that when we have two triangles ABC and ABC', then the number of crossings made by a polygonal path against the shared edge AB is always the same. This is not obvious from our formulation, because the number of crossings at AB is dependent on a choice of triangle with edge AB. We need to show that this choice is arbitrary.

Now the definition of a crossing makes use of a triangle's interior, and different triangles sharing the edge AB will have interiors which may be disjoint, may overlap, or may contain one another. We will need to verify that, nevertheless, the values of the function `crossing` are always consistent. In other words, we must show that the expression "crossings at AB" is well-defined, without reference to the vertex C.

There is key case-split here, shown in Figure 10.19. If C and C' are on the same side of AB as in case (a), then the triangle interiors will overlap. Here, as we cross the edge AB, we enter or leave the interiors of both triangles together, a fact we verify as Theorem 10.13 in Figure 10.20. On the other hand, if C and C' are on opposite sides of AB as in (b), then the interiors of the two triangles are disjoint. In this case, as we move from the interior of ABC across the edge AB to the exterior, we simultaneously

$$\begin{aligned}
&\vdash \texttt{on_line}\ A\ (\texttt{line_of_half_plane}\ hp) \wedge \texttt{on_line}\ B\ (\texttt{line_of_half_plane}\ hp) \\
&\quad \wedge \texttt{on_half_plane}\ hp\ C \wedge \texttt{on_half_plane}\ hp\ C' \\
&\quad \wedge \texttt{between}\ A\ P_i\ B \wedge \neg\texttt{between}\ A\ P_{i+1}\ B \\
&\qquad \Longrightarrow \left(\begin{array}{l} (\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in_triangle}\ (A,B,C)\ R) \\ \quad \Longleftrightarrow\ (\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in_triangle}\ (A,B,C')\ R) \end{array} \right)
\end{aligned} \tag{10.13}$$

Figure 10.20: Moving across AB when C and C' are on the same side.

move from the exterior of ABC' to the interior, a fact we verify as Theorem 10.14 in Figure 10.21.

These two theorems are proven by reasoning about half-planes and, in both cases, applying Theorem 10.7. The assumptions on half-planes in Theorem 10.13 require that the points C and C' lie on the same side of AB. In Theorem 10.14, they require that C and C' lie on opposite sides. Theorem 10.14 needs some extra assumptions since the negations make for very weak claims. For instance, the fact that C' does not lie on hp might just mean that it lies on the line AB, so we have to add in a condition that the points A, B and C' are non-collinear.

The important assumption to note in both theorems is `between` $A\ P_i\ B$. This reflects the fact that the case-split is only pertinent when we come to update and make use of the context variable Γ, which happens when the edge P_iP_{i+1} has exactly one endpoint on the segment AB. So when the edge *lands* on AB, the context variable must be correctly updated to say that we were last inside or outside the triangle. When it *emerges* from AB, the context variable must be correctly utilised to say whether the edge P_iP_{i+1} counts as a crossing. These matters can be formally clarified by the corollaries in Figures 10.22 and 10.23 (we do not reproduce the assumptions in full).

Thus, when C and C' are on the same side, we expect the context values to be the same when we hit the segment AB, and we expect them to compute the same crossing value when we leave AB. When C and C' are on opposite sides, we expect the context values to be opposite when we hit AB, and, as before, we expect them to compute the same crossing value when we leave the segment.

In any case, we know that the vertex C in the expression `crossing` $(A,B,C)\ \Gamma\ P_i\ P_{i+1}$ can be varied about the sides of AB (so long as we change Γ appropriately), and so we

$$\begin{aligned}
&\vdash \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C'\ a) \\
&\quad \wedge \neg\texttt{on_polypath}\ [P_i, P_{i+1}]\ A \wedge \neg\texttt{on_polypath}\ [P_i, P_{i+1}]\ B \\
&\quad \wedge \texttt{on_line}\ A\ (\texttt{line_of_half_plane}\ hp) \wedge \texttt{on_line}\ B\ (\texttt{line_of_half_plane}\ hp) \\
&\quad \wedge \texttt{on_half_plane}\ hp\ C \wedge \neg\texttt{on_half_plane}\ hp\ C' \\
&\quad \wedge \texttt{between}\ A\ P_i\ B \wedge \neg\texttt{between}\ A\ P_{i+1}\ B \\
&\qquad \Longrightarrow \left(\begin{array}{l} (\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in_triangle}\ (A,B,C)\ R) \\ \quad \Longleftrightarrow\ \neg\exists R.\ \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in_triangle}\ (A,B,C')\ R \end{array} \right)
\end{aligned} \tag{10.14}$$

Figure 10.21: Moving across AB when C and C' are on opposite sides.

$$\begin{aligned}
&\vdash \neg\texttt{between}\ A\ P_i\ B \wedge \texttt{between}\ A\ P_{i+1}\ B \\
&\qquad\qquad \Longrightarrow \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} \\
&\quad \vdash (\texttt{between}\ A\ P_i\ B \Longrightarrow \Gamma = \Gamma') \\
&\qquad \Longrightarrow \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \texttt{crossing}\ (A,B,C')\ \Gamma'\ P_i\ P_{i+1}
\end{aligned}$$

Figure 10.22: Well-definedness theorems when C and C' are on the same side of AB

$$\begin{aligned}
&\vdash \neg\texttt{between}\ A\ P_i\ B \wedge \texttt{between}\ A\ P_{i+1}\ B \\
&\qquad\qquad \Longrightarrow \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \neg\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} \\
&\quad \vdash (\texttt{between}\ A\ P_i\ B \Longrightarrow \Gamma = \neg\Gamma') \\
&\qquad \Longrightarrow \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \texttt{crossing}\ (A,B,C')\ \Gamma'\ P_i\ P_{i+1}
\end{aligned}$$

Figure 10.23: Well-definedness theorems when C and C' are on opposite sides of AB

can generalise our notion of crossing. Instead of saying that a polygonal path crosses the side of a *triangle* by moving from interior to exterior and *vice versa*, we can say that a polygonal path crosses an arbitrary *segment* precisely when it moves from one side of the segment to the other. In other words, we can abstract away the vertex C. The mechanics of this will become clear in our final proof in §10.6.1. Before we get to that, we must consider how we initialise Γ.

10.5.4 Initialising the Context

Recall that, in general, when a segment P_iP_{i+1} emerges from the boundary of a triangle ABC, the question of whether there is a crossing depends on additional information provided by the context.

When computing the total crossings for a polygonal path, this context is threaded through the calculations for each individual edge, starting from some initial context. The question is: how do we choose this initial context?

Sometimes, the answer is straightforward. If the endpoint P_i lies in the interior of the triangle, then the value of the context used to compute crossings for P_iP_{i+1} *must* be $\top$. If P_i lies in the exterior of ABC, then the value *must* be $\bot$. These give us some solutions for the initial context, and it would be convenient if we could rely on this simple case. But what happens if P_i lies *on* the triangle?

We considered two possible ways to deal with this question, one of which turned out to have a surprising difficulty which left us favouring the other for the final verification. The first approach has us avoid the question, by always counting crossings from a vertex which does not lie on the triangle. The second approach has us compute a starting context based on the points of the polygon.

The first approach could work because if a closed polygon crosses a triangle, it must have at least one vertex outside the triangle. We just need to verify this and then rotate the polygon's vertex list so we can start counting crossings from there, and we will show how to perform such a rotation in §11.5.1.

However, consider what happens as we vary the point C of the triangle, something we need to do during the sketch proof. An example is shown in Figure 10.24, where we have two triangles ABC and ABC' sharing an edge AB, each crossed by the polygonal path $P_1P_2P_3P_1$. As we claimed, this polygonal path has a vertex outside of each triangle (P_3 for $\triangle ABC$ and P_1 for $\triangle ABC'$), but there is no vertex off both boundaries. This

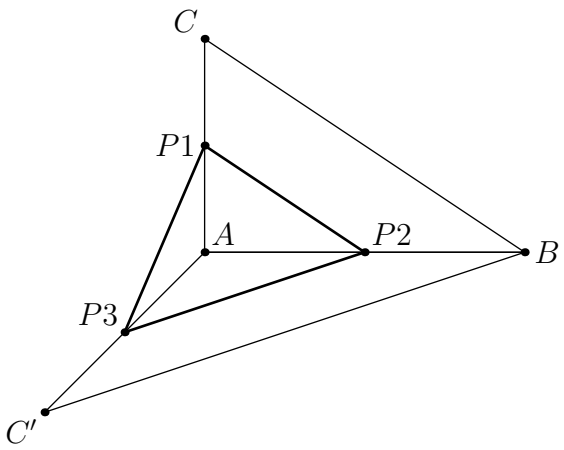


Figure 10.24: No well-defined initial context

means that after we move the point C to C', we would need to perform another polygon rotation. Doing this continually during our proof would complicate the basic inductive argument.

For the second approach, we need to compute a single consistent value for the initial context of any polygon. Fortunately, such a value exists. To spot it, we just realise that the initial value of the context for a polygonal path is related to the value of the context at the polygonal path's final edge. We can compute this final value with a recursive function:

$$\vdash_{def} \Gamma_{final}\ (A,B,C)\ \Gamma\ [] = \Gamma.$$

$$\vdash_{def} \Gamma_{final}\ (A,B,C)\ \Gamma\ ((P_i,P_{i+1})\ ::\ segments) = \Gamma_{final}\ (A,B,C)\ (\Gamma_{next}\ (A,B,C)\ \Gamma\ \ P_i\ P_{i+1})\ segments.$$

Now it turns out that if we push the final context back through the above function, we end up with the same value. Formally, $(\Gamma_{final}\ (A,B,C)\ \Gamma\ segments)$ is a fixpoint of the function $(\lambda\Gamma'.\ \Gamma_{final}\ (A,B,C)\ \Gamma'\ segments)$ for arbitrary Γ. No matter what our starting choice of Γ, the computed final context Γ_{final} can be consistently taken as the initial context from then on. This expression therefore gives us a suitable starting context.

The computation of the initial context appears in our specification of crossings, and the fact that this expression denotes a fixpoint is a lemma used in the verification of Theorem 10.16.

$$\vdash \texttt{polypath\_crossings}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Ps) > 0 \implies \exists Q.\ \texttt{on\_polypath}\ Ps\ Q \wedge\ \texttt{between}\ A\ Q\ B \quad (10.15)$$

$$\begin{aligned}
&\vdash Qs = [P] + Ps + [P] \\
&\quad \wedge \Gamma_{initial} = \Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs) \\
&\quad \wedge \neg\texttt{on_polypath}\ Qs\ A \wedge \neg\texttt{on_polypath}\ Qs\ B \wedge \neg\texttt{on_polypath}\ Qs\ C \\
&\quad \wedge \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C\ a) \\
&\implies \texttt{even} \left(\begin{array}{l} \texttt{polypath_crossings}\ (A,B,C)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \\ +\ \texttt{polypath_crossings}\ (A,C,B)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \\ +\ \texttt{polypath_crossings}\ (B,C,A)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \end{array} \right)
\end{aligned} \quad (10.16)$$

$$\begin{aligned}
&\vdash Qs = [P] + Ps + [P] \\
&\quad \wedge \neg\texttt{on_polypath}\ Qs\ A \wedge \neg\texttt{on_polypath}\ Qs\ B \\
&\quad \wedge \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C\ a) \\
&\quad \wedge \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C'\ a) \\
&\implies \exists \Gamma'.\ \texttt{polypath_crossings}\ (A,B,C)\ (\Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs)) \\
&\qquad\qquad (\texttt{adjacent}\ Qs) \\
&\qquad = \texttt{polypath_crossings}\ (A,B,C')\ (\Gamma_{final}\ (A,B,C')\ \Gamma'\ (\texttt{adjacent}\ Qs)) \\
&\qquad\qquad (\texttt{adjacent}\ Qs)
\end{aligned} \quad (10.17)$$

Figure 10.25: Final specification of crossings

10.5.5 The Specification of Crossings

At last, we will recover the intuitive idea behind crossings from the mess of case-analyses and implementation detail of the previous sections. In Figure 10.25 we give the key theorems which subsume the important details of the other theorems considered thus far. It is these theorems which we shall appeal to exclusively in §10.6.1, where we verify Veblen's Lemma from §8.4.1.

The first theorem (10.15) is mostly a convenience. It simply relates crossings to intersections, telling us that if there are crossings at AB by a polygonal path Ps, then Ps really does intersect AB. The converse does not hold, since the polygonal path might merely intersect and then "bounce off", thus staying on the same side of AB.

Theorem 10.16 assumes that we have a polygon Qs and sets an initial context as described in the previous section. It also assumes we have a triangle ABC and that Qs does not intersect any of its vertices, as per our discussion in §10.2.3. Under these conditions, the total number of crossings against the three sides is always even.

Theorem 10.17 tells us that the choice of C when counting crossings is arbitrary, so long as it is not on the line AB. There is a slight complication, in that the theorem tells us to reset the initial context using the supplied Γ' given in the conclusion, but since Theorem 10.16 holds for arbitrary choices of Γ, we can ignore this constraint when we apply the two theorems.

The upshot of Theorem 10.17 is that we can understand a crossing without reference to a triangle, but instead only with reference to the points A and B. In the next section, we shall see how this more general understanding plays out in our verification.

10.6 Verifying the Sketch Proof

In this section, we shall review our verification of the parity proof that we sketched in §10.1. There are interesting details in the verification relating to our use of the theorems of the previous section. But more importantly, we can obtain a beautiful theorem from which the first half of the Polygonal Jordan Curve Theorem arises as a corollary. Unlike the Polygonal Jordan Curve Theorem, this theorem makes no reference to *simple polygons*. It is a general theorem about arbitrary polygonal paths, one which does not hinge on complex definitions such as those that appear for the Polygonal Jordan Curve Theorem.

$$\begin{aligned}
&\vdash \texttt{length}\ Qs \geq 2 \wedge \texttt{head}\ Qs = \texttt{last}\ Qs \wedge P_1 \neq P_2 \\
&\quad \wedge \left(\begin{array}{l} \forall C.\ \neg(\exists a.\ \texttt{on_line}\ P_1\ a \wedge \texttt{on_line}\ P_2\ a \wedge \texttt{on_line}\ C\ a) \\ \qquad\qquad \Longrightarrow\ \exists \Gamma.\ \texttt{odd}\ (\texttt{polypath_crossings}\ (P_1, P_2, C) \\ \qquad\qquad\qquad\qquad (\Gamma_{final}\ (P_1, P_2, C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs)) \end{array} \right) \\
&\quad \Longrightarrow\ \exists X.\ \texttt{on_polypath}\ ([P_2] + Ps + [P_1])\ X \wedge \texttt{on_polypath}\ Qs\ X.
\end{aligned} \tag{10.18}$$

Figure 10.26: Parity Lemma

10.6.1 The Induction Proof

The parity proof assumes that we have two polygons Ps and Qs intersecting at an edge. Based on this, we consider a sequence of triangles formed from the vertices of the polygon Ps, and repeat a parity argument over the number of crossings.

This argument readily formalises as a proof by induction, which gives us a nice reinterpretation. Rather than considering triangles with vertices drawn from Ps, we continually reduce the problem to smaller polygons. This inductive proof yields the (somewhat ugly) lemma given in Figure 10.26.

Here, we assume two polygons of length at least two, namely $[P_1, P_2] + Ps + [P_1]$ and Qs. The polygon Qs is assumed to cross the edge P_1P_2 an odd number of times. We then conclude that Qs intersects the tail of $[P_1, P_2] + Ps + [P_1]$, exactly as we require in the sketch proof.

Of particular note is how we formalise the idea that Qs crosses the edge P_1P_2 in terms of the function `polypath_crossings`: we abstract away the C and the Γ variables with universal and existential quantifiers, knowing it is valid to do so based on our well-definedness theorems.

Using structural induction on the list Ps, we become tasked with showing that the polygonal path Qs crosses the path $[P_2, P_3] + Ps' + [P_1]$ for any P_3 and Ps' where Ps' satisfies the inductive hypothesis. To do this, we show that the number of crossings at P_1P_3 is odd, and then apply the induction hypothesis, to tell us that Qs crosses the path $[P_3] + Ps' + [P_1]$. The desired result then follows.

We assume here that Qs does not cross at P_2P_3, since otherwise we are done. In Mizar Light, this assumption is made quite literally:

$$\boxed{\texttt{assume}\ \neg(\exists X.\ \texttt{on\_polypath}\ [P_2,P_3]\ X \wedge \texttt{on\_polypath}\ Qs\ X)}$$

Now according to our treatment of the idea of crossings from §10.5.3, our goal is formalised as

$$\begin{aligned}&\forall C.\ \neg(\exists a.\ \texttt{on\_line}\ P_1\ a \wedge \texttt{on\_line}\ P_3\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad\Longrightarrow\ \exists \Gamma.\ \texttt{odd}\ (\texttt{polypath\_crossings}\ (P_1,P_3,C)\\ &\qquad\qquad (\Gamma_{final}\ (P_1,P_3,C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs)).\end{aligned}$$

There is actually a case-split to consider here. It is possible that P_3 lies on the line of P_1P_2, or, more specifically, on the *ray* $\overrightarrow{P_1P_2}$[1]. We shall not cover the details of this case. Suffice to say, it requires a complication of Theorem 10.17, which we give in Appendix C. Explaining it here would just obscure the basic ideas of the verification.

Thus, we shall assume that P_3 forms a triangle with P_1P_2. This means we can apply our assumption that there are an odd number of crossings at P_1P_2, namely

$$\begin{aligned}&\forall C.\ \neg(\exists a.\ \texttt{on\_line}\ P_1\ a \wedge \texttt{on\_line}\ P_2\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad\Longrightarrow\ \exists \Gamma.\ \texttt{odd}\ (\texttt{polypath\_crossings}\ (P_1,P_2,C)\\ &\qquad\qquad (\Gamma_{final}\ (P_1,P_2,C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs))).\end{aligned}$$

We now have an idea of how the quantifiers in this formula are to be used. We need C to be arbitrary, because we must instantiate it with the particular vertex P_3 that we have obtained from the list Ps. We must then obtain an appropriate starting context Γ depending on C, which is chosen according to which side of the edge P_1P_2 the vertex P_3 lies.

By applying Theorems 10.15 and 10.16, we can then conclude that there must be an odd number of crossings at P_1P_3 *with respect to the triangle* $P_1P_2P_3$. All we need now to complete the inductive step is to generalise this claim by quantifying over the variable P_2. For this final step, we just use our well-definedness theorem (10.17).

10.6.2 A Theorem of Polygonal Paths

Theorem 10.18 is all we need to show that a simple polygon divides the plane into two regions. It is interesting to note, however, that we have not mentioned path connectedness in this theorem, nor have we supposed that the polygons concerned are simple.

[1] For this inference, we use our linear reasoning tactic.

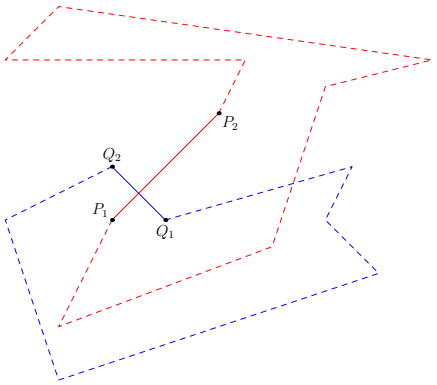


Figure 10.27: Arbitrary intersecting polygons

This suggests there is a more general corollary to be had, one whose formalisation does not mention *crossings* at all. Indeed, this ugly concept serves only as the crucial scaffolding of the verification. It is absent in the statement of the final theorem.

In a fairly short verification (42 steps), we apply Theorem 10.18 to obtain a beautifully symmetric theorem concerning arbitrary intersecting polygons:

$$\begin{aligned}
&\vdash \neg(\exists a.\ \texttt{on_line}\ P_1\ a \wedge \texttt{on_line}\ P_2\ a \wedge \texttt{on_line}\ Q_1\ a) \\
&\quad \wedge \neg(\exists a.\ \texttt{on_line}\ Q_1\ a \wedge \texttt{on_line}\ Q_2\ a \wedge \texttt{on_line}\ P_1\ a) \\
&\quad \wedge \texttt{between}\ P_1\ X\ P_2 \wedge \texttt{between}\ Q_1\ X\ Q_2 \\
&\quad \wedge P_1 = \texttt{last}\ Ps \wedge Q_1 = \texttt{last}\ Qs \\
&\quad \Longrightarrow \exists Y.\ \texttt{on_polypath}\ (P_2 :: Ps)\ Y \wedge \texttt{on_polypath}\ (Q_1 :: Q_2 :: Qs)\ Y \\
&\qquad\qquad \vee \texttt{on_polypath}\ (P_1 :: P_2 :: Ps)\ Y \wedge \texttt{on_polypath}\ (Q_2 :: Qs)\ Y.
\end{aligned} \tag{10.19}$$

In words, if we have polygons $P_1P_2\ldots P_1$ and $Q_1Q_2\ldots Q_1$ such that the segment P_1P_2 and Q_1Q_2 intersect, then one of the polygons intersects a non-trivial suffix of the other.

The theorem places no constraints on the polygons other than that they cross at their first edges. They can have repeated vertices; they can self-intersect; they could even be the trivial polygons $P_1P_2P_1$ and $Q_1Q_2Q_1$. The point to visualise is that if two segments P_1P_2 and Q_1Q_2 cross one another, and we attempt to connect P_2 back to P_1 whilst attempting to connect Q_2 back to Q_1, we will find another point of intersection. See Figure 10.27.

10.7 The Plane Divides into at Least Two Regions

We can now verify the main theorem for this chapter. We assume a simple polygon $P_1P_2\ldots P_n$, and we must find two points off this polygon which cannot be connected by a polygonal path without crossing the simple polygon. Equivalently, any polygonal path connecting the two chosen points must intersect the simple polygon.

The strategy we use to prove this has already been covered in §10.1, and the verification respects the structure. In the sketch proof, we consider two rays emerging on either side of the edge P_1P_2. We find the points where these rays intersect *Ps*, and pick the point of intersection closest to *AB*. In our verification, this step is handled by a "ray-casting" theorem which we discuss in §11.3.1.

Ray-casting is the one and only place where we need to assume the simplicity of the simple polygon. In fact, we do not need to assume that much. All we really need to know is that there is *some* edge of a polygon, and *some* point *P* inside that edge, such that *P* does not lie on the rest of the polygon. Under these circumstances, we know that the polygon divides the plane into at least two regions.

This is to be contrasted with the verification in the next section. There, the assumption of a polygon's simplicity will feature heavily. The reason is that there are many ways for a polygon to divide the plane into multiple regions, but fewer ways for a polygon to restrict the number of regions to *two*.

We have come this far by building a large tower of abstractions, complicated and unwieldy definitions, and theorems containing an irritatingly large number of hypotheses. The pay-off from this sort of verification is a result which throws out the scaffolding and brings us to a neat and easily grasped theorem.

$$\begin{aligned}
&\vdash \texttt{simple_polygon}\ \alpha\ Ps \\
&\quad \Longrightarrow\ \exists P.\ \exists Q.\ \texttt{on_plane}\ P\ \alpha \wedge \texttt{on_plane}\ Q\ \alpha \\
&\qquad \wedge \neg \texttt{on_polypath}\ Ps\ P \wedge \neg \texttt{on_polypath}\ Ps\ Q \\
&\qquad \wedge \neg \texttt{polypath_connected}\ \alpha\ (\texttt{on_polypath}\ Ps)\ P\ Q.
\end{aligned} \tag{9.3}$$

Chapter 11

Verifying the Polygonal JCT: Part II

We are nearing the end of our verifications. All that remains is to verify the second half of the Jordan Curve Theorem for polygons based on the axioms of Hilbert's ordered geometry. In this half of the verification, we must prove that a simple polygon separates its plane into at most two regions. As discussed when we gave the formalisation of this theorem in Chapter 9, it amounts to proving that given three points in the plane and not on the polygon, at least two of them are connected by a polygonal path.

This is effectively a maze navigation problem, lively and visual, with lots of geometrically interesting lemmas. Unlike the "crossings" of the last chapter, the basic concepts we appeal to are reflected cleanly in the low-level details of the verification, rather than being obscured by case-analyses and edge cases.

In discussing our verification, we will cover the same basic ground as we did in the last chapter. In §11.1, we shall lay out the general approach of the proof. In §11.2, we will look more closely at some of the basic machinery we will need, and formalise the key concepts in higher-order logic. Then, in §11.3, we shall cover the key lemmas that support our basic formalised concepts. As in the last chapter, these lemmas can be divided into those which introduce points in a geometrical configuration, and those which allow us to infer properties of the resulting configurations.

In the rest of the chapter, we shall look in more detail at how the lemmas are applied to recover all the details of the sketch proof. We provide a few readable extracts of interesting verifications, demonstrating how faithfully we can formalise the intuitive synthetic arguments.

11.1 Strategy

The basic intuition behind the proof is similar to the ones presented by Veblen [100] and Feigl [24]. We follow Veblen's proof the most closely. Contrary to Guggenheimer's [27] claim that Veblen's proof only holds for convex polygons, we believe the evidence of this chapter shows that Veblen was basically correct. That said, our verification is based on a more thorough analysis than presented by Veblen.

We are required to show that, given three points in the plane and not on a polygon, at least two of them are connected by a polygonal path. Let us reinterpret this and understand the three points as three players trying to navigate a polygonal maze.

Our basic goal is to get the three players "next to" the same edge. Then we just need to find a path between whichever of the two players are on the same side of that edge. We will find that the difficulty here lies in getting the players through potentially very tight corridors, and around difficult corners. We must show how to obtain paths for the players without recourse to notions such as comparable directions, parallel lines or distances. This will rule out common approaches to the theorem, such as the one given by Tverberg [97]. In Tverberg's proof, we just need to consider a sufficiently small region around the edges of the maze (an "offset curve"), which we know to be polygonal path-connected. Without notions of distance, this description is out of scope of Hilbert's ordered geometry.

One way to formulate the idea that the players are "next to" the same edge of the maze is to assert that all three have line-of-sight to that edge. This metaphor does not appear explicitly in Veblen or Feigl's proof, but it can be read into both, and we found it extremely helpful in providing an intuitive grasp of the formalisation.

To make clear the idea about lines-of-sight, we will depict our players as *eyes*, with a dashed line-of-sight to a point of the maze. In Figure 11.1, we show players *Red*, *Black* and *Blue* situated and staring at various points of a maze. Players *Red* and *Black* are inside the maze, while *Blue* is on the outside. In the figure, we depict the paths they follow as they traverse the maze so that they have line-of-sight to the edge P_iP_{i+1}. Since *Red* and *Black* end up on the same side of P_iP_{i+1}, we can connect them by a polygonal path.

The paths we have drawn through the maze are potential witnesses to the paths we consider in our verification. In fact, we can take the line-extension axiom

$$\vdash A \neq B \implies \exists C.\, \texttt{between}\, A\, B\, C \qquad \text{(II, 2)}$$

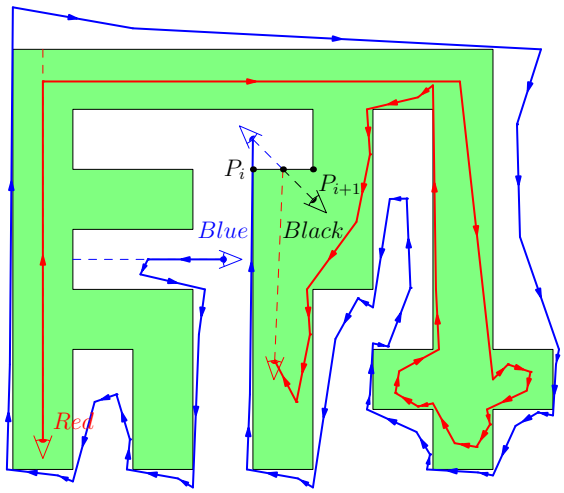


Figure 11.1: Navigating a maze

and suppose that the witness C in the conclusion is always chosen so that B is half-way between A and C. In this case, the paths sketched in Figure 11.1 are precisely those that we witness in our formal verification.

11.2 Formulation and Formalisation

Compared to the last chapter, where we introduce the complex idea of a crossing, the basic ideas needed in the verification for this chapter are relatively straightforward. Firstly, given a simple polygon Ps, we will say that a point X has line-of-sight to a point X' if there is no point of Ps which lies strictly between X and X'. We shall say that when the point X' lies between vertices P_i and P_{i+1} of Ps, then the point X has line-of-sight to the *edge* P_iP_{i+1}. The situation is formalised as

$$\neg \texttt{on\_polypath}\ Ps\ X \wedge \texttt{between}\ P_i\ X'\ P_{i+1}$$
$$\wedge \neg \exists Z.\ \texttt{between}\ X\ Z\ X' \wedge \texttt{on\_polypath}\ Ps\ Z.$$

Our verification breaks down into three parts. Firstly, we must show how every point not on a simple polygon has a line-of-sight to some edge of the maze. Next, we must show how, if a point X has line-of-sight to an edge P_iP_{i+1}, then there is a polygonal path to a point Y which has line-of-sight to the next edge $P_{i+1}P_{i+2}$. As such, for any edge and any point X, there is a polygonal path from X to another point which has line-of-sight to that edge. Finally, we must show that if two players have line-of-sight to the same edge, and lie on the same side of that edge, then there is a polygonal path between them.

We shall describe the informal proofs and verifications of each part in §11.4. First, we consider the crucial supporting lemmas.

11.3 Obtaining Lines-of-Sight

We have two key theorems: a ray-casting theorem which obtains a new line-of-sight, and a theorem dubbed "squeeze" which handles narrow cracks in corridors. In proving the squeeze theorem, we shall need recourse to many of our earlier theorems about triangles and their interiors, and a new theorem about a triangle containing another triangle.

11.3.1 Ray-casting

Our ray-casting theorem gives us a line-of-sight to a polygonal path, aimed in an arbitrary direction towards that path. To achieve this, we must find the first point of intersection that the ray makes with the polygonal path. Ray-casting is actually needed in the first half of the Polygonal Jordan Curve Theorem (see §10.7), but it makes more sense to explain it here where we are appealing to metaphors from computer graphics.

Ray-casting appears in a weakened form in Veblen's proof, but there he only considers casting a ray which does not intersect any *vertex* of the polygonal path. This can be generalised by considering a few additional cases, after which we have a much more useful theorem.

This ray-casting theorem relies almost exclusively on linear reasoning and we found it particularly tricky to verify. As with our verification of Theorem 10.12 in the last chapter, our linear reasoning tactic came through as a powerful tool for dealing with these problems, but first it has to be fed the right starting hypotheses. The trouble we had in the verification was deciding which hypotheses were needed.

It often ended up being a matter of trial and error, but fortunately, the linear reasoning tactic is based on a decision procedure. When there were not enough facts available, it would promptly terminate and announce that the goal was not solvable. We could then go back through the problem and try to identify additional facts to feed the tactic and then retry.

The need for this sort of manual labour is not particularly worthy of an automated proof assistant, and in future, some feedback on why the tactic failed would be useful

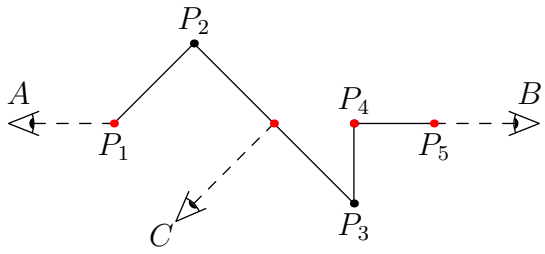


Figure 11.2: Ray-casting

and reasonably straightforward to implement. We are only working here with small sets of points (and the tactic struggles anyway when larger sets are considered), so it is feasible to enumerate all their permutations, including permutations when some combination of the points are equal. For instance, in the case of 5 points, there are only 431 possible arrangements. These serve as models, which can be filtered down by finding those which satisfy both the current hypotheses and the negation of the conclusion. After filtering, we are left with just the counterexamples, which would help if we could identify features in them which we can recognise as impossible given our diagrams and general intuition about the proof. We leave such a counterexample-checker for future work.

Sometimes, such counterexamples mean that a case-split must be considered. Sometimes, this reflected two different linear reasoning problems, which meant providing two subproofs. But sometimes we got lucky. If the case-split led to a single linear reasoning problem, we could let our incidence-discoverer handle the case-analyses automatically using its internal representation of proof trees.

The verification applies structural induction on the vertex list, and reduces the problem to that of ray-casting to a single edge. The basic case-analyses are shown in Figure 11.2. We cast rays from the points A, B and C to the polygonal path $P_1P_2 \ldots P_5$. The salient differences between the three lines-of-sight are as follows: point A has line-of-sight to an endpoint of an edge, but the edge itself is not on the line-of-sight. Point B has line-of-sight to an endpoint of an edge, and the edge itself *is* on the line-of-sight. Finally, point C has line-of-sight to the interior of an edge P_2P_3.

We now give the formalisation of the theorem. From a point X, we fire a ray to an arbitrary point P on the polygonal path, and then obtain a point Y to which X has line-of-sight. Though it does not prove necessary in our verifications, we provide some extra information about the point Y, namely that it is either strictly between X and P,

or else we already had line-of-sight to P.

$$\vdash \neg\texttt{on\_polypath}\ Ps\ X \wedge \texttt{on\_polypath}\ Ps\ P \implies \left(\begin{array}{c} \exists Y.\ \texttt{on\_polypath}\ Ps\ Y \wedge (\texttt{between}\ P\ Y\ X \vee P = Y) \\ \wedge \neg(\exists Q.\ \texttt{between}\ X\ Q\ Y \wedge \texttt{on\_polypath}\ Ps\ Q) \end{array}\right). \quad (11.1)$$

Note the first conjunct in the hypothesis of this theorem. We can only cast rays to a polygonal path if we are not on that path. This should clarify a point made in §10.7 of the last chapter. There, we said that the final part of the verification showing that there are at least two regions of a simple polygon is based on ray-casting from some point X on that polygon. In particular, we are ray-casting to the rest of the polygon, and thus, if we are to ray-cast, we need to assume that the rest of the polygon does not have a self-intersection at X. This we guarantee based on the fact that the polygon is assumed to be simple.

11.3.2 Squeeze

The most powerful theorem in our arsenal is one we dubbed "squeeze", since the intuition is that it allows us to find segments which squeeze through arbitrarily narrow gaps of a maze. What counts as a narrow gap in the abstract world of ordered geometry is determined by the betweenness relation, and so the basic axioms governing this relation limit our powers in navigating such gaps. We can get some idea of the challenge by realising that, on some interpretations, these gaps are *infinitesimally* narrow. Hilbert's axioms are independent of Archimedes' axiom.

Abstractly, our *squeeze* theorem, Theorem 11.2, tells us that if we have a polygonal path $[A,B,C]$ which is not intersected by the polygonal path Ps, then we can introduce a point A' between A and B such that Ps intersects $A'C$ in at most one place. We can apply and interpret this theorem in a number of ways. In §11.3.3, we will show how to interpret it in terms of lines-of-sight. Here, we interpret it in terms of finding diagonals that divide a polygon into two simple polygons.

$$\begin{array}{l} \vdash \neg\texttt{on\_polypath}\ Ps\ B \\ \quad \wedge \neg(\exists X.\ \texttt{between}\ A\ X\ B \wedge \texttt{on\_polypath}\ Ps\ X) \\ \quad \wedge \neg(\exists X.\ \texttt{between}\ B\ X\ C \wedge \texttt{on\_polypath}\ Ps\ X) \\ \quad \implies \exists A'.\ \texttt{between}\ A\ A'\ B \wedge \neg\exists X.\ \texttt{in\_triangle}\ (A',B,C)\ X \wedge \texttt{on\_polypath}\ Ps\ X. \end{array} \quad (11.2)$$

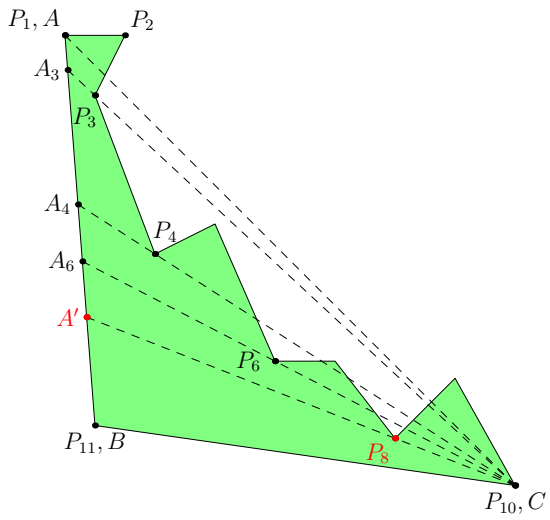


Figure 11.3: Squeezing a diagonal

In Figure 11.3, we use squeeze to find a diagonal of the polygon $P_1P_2 \ldots P_{11}$. Here, we have set $A = P_1$, $B = P_{11}$, and $C = P_{10}$, while we set *path* to be the rest of the polygon $P_1P_2P_3 \ldots P_{10}$.

The form of the conclusion in Theorem 11.2 reflects its verification. We make a slightly different claim than declaring the existence of a diagonal. We say instead that the polygonal path does not lie in the interior of $\triangle A'BC$, which means that any point between A' and B yields a segment with the point C which does not intersect Ps. We prove this starting with the triangle ABC, and find the first vertex in Ps which lies inside this triangle. In the case shown, this would be the vertex P_3. We draw a line through P_{10} and P_3 to the point A_3, and continue the argument with this new triangle. Eventually, we will be left with a triangle whose interior contains no point of Ps. In a sense, the triangle ABC has been "squeezed" by Ps into the triangle $A'BC$.

Unhappily, proving that $\triangle A'BC$ contains no point of Ps has us boiled down in case-splits similar to those needed to analyse triangle crossings in the last chapter (§10.5). Rather than go into the details of these, we shall focus on the more illuminating verification that $\triangle A'BC$ contains no *vertex* of Ps. This only boils down to two more lemmas for triangle interiors.

11.3.2.1 Another "Inner Pasch" Rule

Our first supporting theorem allows us to introduce the intersection points A_3, A_4, A_6 and A' in Figure 11.3. This theorem is similar in spirit to the Pasch axioms (5.2, 5.3)

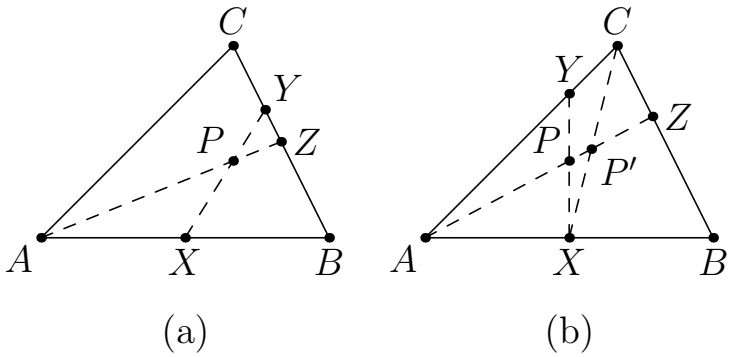


Figure 11.4: Drawing a line from a vertex to the opposite side

and its variant for triangle interiors (10.6). It has a very succinct formalisation, but a non-trivial verification:

$$\vdash \texttt{in\_triangle}\,(A,B,C)\,P \implies \exists X.\,\texttt{between}\,B\,X\,C \wedge \texttt{between}\,A\,P\,X. \qquad (11.3)$$

The verification is the most interesting for these point introduction theorems. Unlike the verification of Theorem 10.6 from the last chapter, we find ourselves back employing Pasch's axiom (II, 4) rather than exploiting our theorems of half-planes. First, we use THEOREM 3 to find a point X on AB. Next, we apply (II, 4) to the triangle ABC and the line PX. This gives us a point Y which is either between B and C or between A and C. Here is one of the rare times where both cases are possible. Normally, we would be able to refute one of the cases based on incidence reasoning. Here, we need two subproofs.

If Y lies between B and C as in case (a) of Figure 11.4, then we apply (II, 4) to the triangle BXY and the line AP to find a point Z between B and Y. This point is then between B and C (via linear reasoning on B, C, Y and Z). Furthermore, P is between A and Z by (10.9).

In case (b) of Figure 11.4, we find that P is now an interior point according to Veblen's definition and the position of the points X and Y. Here, we apply (II, 4) to the triangle CXY and the line AP to find a point P' on CX. By the same axiom applied to $\triangle BCX$ and the line AP', we find the desired point Z on BC.

11.3.2.2 Subtriangles

In our verification of Theorem 11.2, we consider a sequence of triangles such as those of Figure 11.3, namely $\triangle ABC$, $\triangle A_3BC$, $\triangle A_4BC$, $\triangle A_6BC$ and $\triangle A'BC$. Each of these

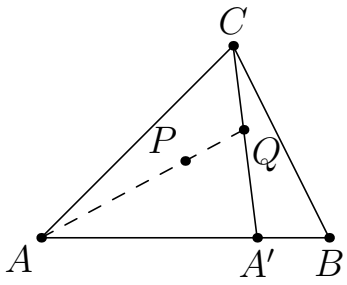


Figure 11.5: Any interior point P of $\triangle AA'C$ is an interior point of $\triangle ABC$.

is intended to exclude one vertex of *Ps* from its interior. By the time we reach the last triangle, we can conclude that all vertices of *Ps* will lie outside the interior. This conclusion assumes a transitivity property, which is given by our second lemma. It shows that the ordering of A, A_3, A_4, A_6 and A' along the segment AB yields a sequence of triangles with *nested* interiors.

The verification, given in Figure 11.6, is extremely short and readable, making use of a number of lemmas we have previously considered, including Theorem 11.3 from the last subsection. We show that any point P interior to a triangle $AA'C$ is interior to any triangle ABC where A' is between A and B. That is, the interior of $\triangle AA'C$ is nested in $\triangle ABC$. To prove it, we put a point Q on the line $A'C$, and show that it must be interior to $\triangle ABC$ on the basis of Theorem 10.8. It then follows by Theorem 10.9 that P must also be interior. See Figure 11.5.

```
theorem in_triangle (A,A',C) P ∧ between A A' B ⟹ in_triangle (A,B,C) P
assume in_triangle (A,A',C) P ∧ between A A' B                              0
so consider Q such that between A' Q C ∧ between A P Q by (11.3)            1
obviously (by_ncols ∘ add_in_triangle ∘ conjuncts)
    hence in_triangle (A,B,C) Q by (10.3),(10.8),(II, 1) from 0
qed from 1 by (10.4),(10.9)
```

Figure 11.6: Subtriangles

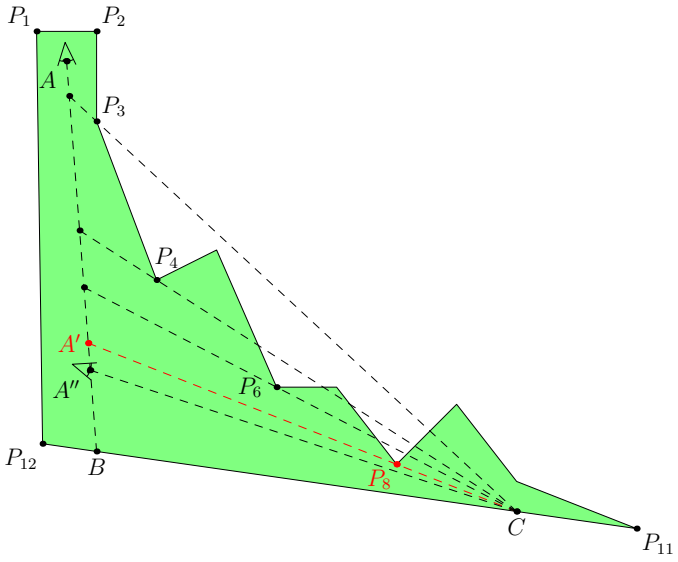


Figure 11.7: Obtaining line-of-sight to C

11.3.3 Obtaining lines-of-sight via Squeeze

As we suggested earlier, Theorem 11.2 is quite powerful. In this section, we will show two applications of the theorem in terms of lines-of-sight, both of which we shall need for the core verifications of this chapter.

11.3.3.1 Moving to a new line-of-sight

Consider Figure 11.7. Here, we have slightly modified and relabelled the diagram from Figure 11.3 so that we can interpret the point A as our player's starting location, with line-of-sight to a point B on the edge $P_{11}P_{12}$ of a simple polygon. The goal is to obtain line-of-sight to a given point C on that same edge.

The idea is that if the player proceeds far enough down their initial line-of-sight, we can easily rotate them to the new line-of-sight. We do this by applying Theorem 11.2 and taking *path* to be the fragment of the polygon $P_{12}P_1P_2\ldots P_{11}$. Let us think about how to fulfil the hypotheses. We are basically being asked to rule out the possibility that the polygonal path $[A,B,C]$ is not intersected by this fragment, save for possible intersections at A and at C.

Well, we know that the segment AB does not lie on the fragment $P_{12}P_1P_2\ldots P_{11}$, because AB is supposed to be a line-of-sight. We also know that BC does not intersect the fragment, because we are assuming *simple* polygons: the polygon should not self-intersect the edge $P_{11}P_{12}$, and this edge contains BC. We leave open the possibility that

$P_{12}P_1P_2\ldots P_{11}$ intersects the lone point C, since, as we shall see in §11.4.1, we will sometimes need to obtain line-of-sight to a vertex, where we will set $C = P_{11}$.

Applying Theorem 11.2 is not quite enough. We now have a point A' which *almost* has line-of-sight to C. We just need to use THEOREM 3, obtaining a point A'' and a segment $A''C$ which lies properly in the triangle $A'BC$, and, as per the conclusion of Theorem 11.2, a segment which is a line-of-sight to the point C. We verify the argument here as a corollary of Theorem 11.2:

$$\begin{aligned}
&\vdash \neg \texttt{on_polypath}\ \mathit{path}\ B \\
&\quad \wedge \neg(\exists X.\ \texttt{between}\ A\ X\ B \wedge \texttt{on_polypath}\ \mathit{path}\ X) \\
&\quad \wedge \neg(\exists X.\ \texttt{between}\ B\ X\ C \wedge \texttt{on_polypath}\ \mathit{path}\ X) \\
&\quad \Longrightarrow \exists A'.\ \texttt{between}\ A\ A'\ B \wedge \neg\exists X.\ \texttt{between}\ A'\ X\ C \wedge \texttt{on_polypath}\ \mathit{path}\ X.
\end{aligned} \tag{11.4}$$

Now because AB is a line-of-sight, we know that AA'' is part of a polygonal path which does not intersect $P_1P_2\ldots P_{11}P_{12}P_1$. What we have exploited here is the fact that we can always move along our lines-of-sight without intersecting the polygon. The use of squeeze (11.2) in this section is therefore telling us how to build up paths through a maze, by getting the player to move far enough along their line-of-sight that they will be able to see a point further down the current edge.

The next application of squeeze will be just as critical for our verifications.

11.3.3.2 Rotating to a new line-of-sight

In Figure 11.8, we have again modified and relabelled the diagram. The idea here is that our player is situated at a point C and initially has line-of-sight to the vertex B. The problem with this scenario is that, in our verifications, we are more concerned about having lines-of-sight to points on a polygon which are *not* vertices (for a start, this forces the player off the line of the edge). So we want our player to rotate their line-of-sight slightly.

Therefore, we apply Theorem 11.2 by reversing the order of A, B and C. The polygonal fragment we are concerned with here is again $P_{12}P_1P_2\ldots P_{11}$, and as before, we have to be sure that the hypotheses of Theorem 11.2 have been met. That is, we must show that the polygonal path $[A,B,C]$ is not intersected by the fragment $P_{12}P_1P_2\ldots P_{11}$.

Again, we know that AB is not intersected by the fragment, because we assume that the polygon is simple and so does not have such self-intersections, and we know that

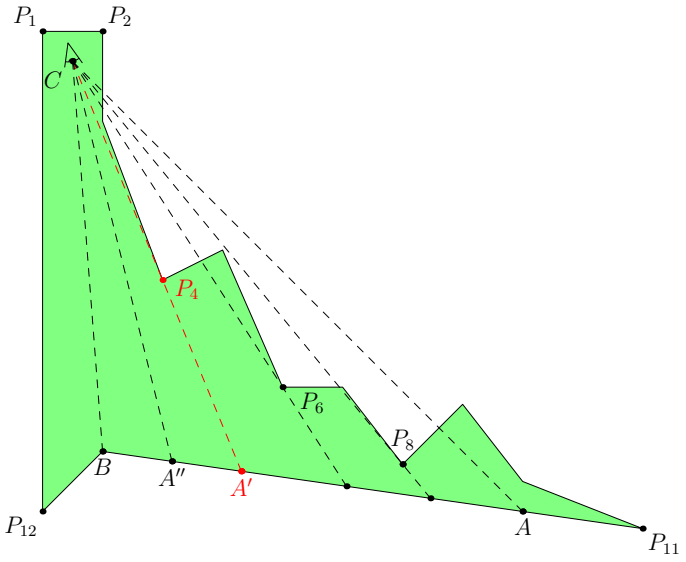


Figure 11.8: Obtaining line-of-sight to the interior of AB

BC is not intersected by the fragment because BC is assumed to be a line-of-sight. This means we have met the hypotheses. We now obtain the point A' as shown, and as before, we use THEOREM 3 to obtain a point A'' between B and A'. Our player will now have line-of-sight to a non-vertex point of the segment AB.

The reader may have noticed that we have not mentioned the segment BP_{12} shown in the diagram. In fact, this segment could pose a problem. If it were chosen to lie on the other side of BC, then it might intersect the segment $A''C$, and if this happens, then $A''C$ will not count as a line-of-sight. Fortunately, we shall be able to rule out this circumstance when we come to apply Theorem 11.4. We shall always be assuming that, when we rotate our line-of-sight, the points P_{11} and P_{12} are on opposite sides of BC, as shown.

To summarise, we can say that the first use of squeeze allows us to reduce the *distance* to the point B sufficiently and thus obtain a new line-of-sight, while this second use of squeeze is telling us that we can reduce the *angle* ABC sufficiently. What the theorem gives us is the ability to reduce distances and angles even without a general theory for comparing them, and without any sort of arithmetic for them. We "squeeze" our distances and angles by working exclusively with a weak order relation and properties of incidence.

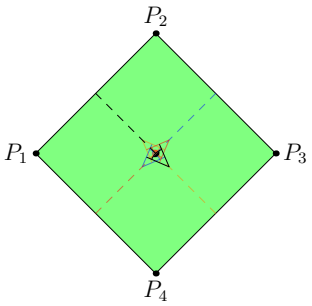


Figure 11.9: Edge-to-edge in a convex polygon

11.4 Edge-to-Edge

Since a simple polygon is just a list of adjacent edges, we can navigate every single edge just by moving between one edge and the next in the adjacency list. In Section 11.2, we explained that the movement from an edge P_iP_{i+1} to an edge $P_{i+1}P_{i+2}$ amounts to showing three things: (1) that there is a point X with line-of-sight to P_iP_{i+1}; (2) that there is a point Y with line-of-sight to $P_{i+1}P_{i+2}$; and (3), that there is a polygonal path between X and Y.

Now when inside a convex polygon, this matter is completely trivial. Indeed, *every* interior point of a convex polygon has line-of-sight to *every* edge, as in Figure 11.9. The salient fact to notice is that, in a convex polygon, interior points are on the same side of every edge as every other vertex.

Generalising this, we can say that two edges P_iP_{i+1} and $P_{i+1}P_{i+2}$ appear "locally convex" from the perspective of a point P if the point P is on the same side of P_iP_{i+1} as P_{i+2}. Otherwise, we can say that the edges appear "locally concave". These provide our two cases for how we navigate the edges of a simple polygon.

11.4.1 Locally Convex Edges

To explain the case for locally convex edges, we shall assume we have a polygon $P_1P_2P_3\ldots P_n$, and we shall assume that we are moving from edge P_1P_2 to edge P_2P_3 (we can use the polygon rotations described in §11.5.1 to consider the other pairs of edges).

In Figure 11.10, we start at a point X which has line-of-sight to P_1P_2. Our goal is to reach a point Y with line-of-sight to P_2P_3. We start by moving towards P_1P_2 by applying Theorem 11.4 as described in §11.3.3.1, seeking a line-of-sight with the vertex P_2.

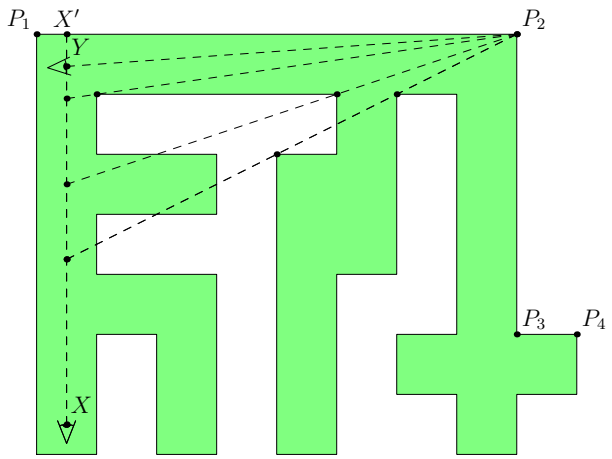


Figure 11.10: Edge-to-edge in a locally convex polygon

Applying squeeze in this way is usually a tricky business, because there are a number of hypotheses which must be fulfilled and it is not always immediately clear how. Even when we had found all the necessary hypotheses, the default `MESON` prover would struggle to apply them all, and so we would have to inject some procedural code as justification. In doing this, we tried to respect the aims of declarative verification. For instance, at no point do we destructively modify the proof context, and we always insist that whenever our tactics apply an assumption, we include the assumption label so that a reader can at least track the dependencies. Here, we use `EXISTS_TAC` followed by `MATCH_MP_TAC`, which leaves `MESON` just having to discharge the assumptions of (11.4).

```
obviously by_incidence so consider Y such that between X Y X'
    ∧∀Z. between Y Z P2 ⟹ ¬on_polypath (P2 :: P3 :: Ps) Y
    from ... by on_polypath, (II, 2)
    using K (MATCH_MP_TAC (11.4) THEN EXISTS_TAC α)                16, 17
```

This done, we have our desired path XY. We know that this path does not intersect any part of the simple polygon, since it is part of our original line-of-sight. All that remains then is to rotate ourselves slightly to obtain a line-of-sight YY' with some point on P_2P_3. This we do in Figure 11.11, using Theorem 11.4 as described in §11.3.3.2.

This is not the whole story. What is missing is any mention of the local convexity. This is needed to show that our segment YY' does not intersect the polygon, and thus is a genuine line-of-sight. Our second application of (11.4) only tells us that it does not intersect the fragment of the polygon $P_3P_4 \ldots P_1$. Our incidence discoverer tells us

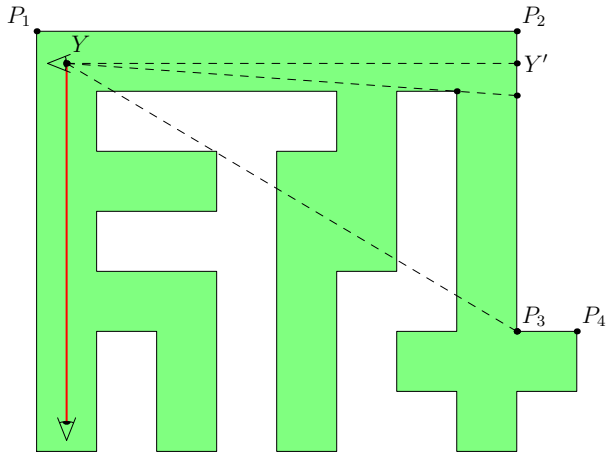


Figure 11.11: Edge-to-edge in a locally convex polygon (continued)

further that any point Z between Y and Y' cannot intersect P_2P_3. What is left to rule out is a possible intersection with P_1P_2. For this, we need to think about half-planes.

We reason as follows. We know that Y and P_3 are on the same side of P_1P_2 (since P_1P_2 and P_2P_3 are locally convex from the perspective of Y), and Y', being on P_2P_3, must also be on the same side of P_1P_2. That means that Z is also on the same side, because it lies on YY'. But that means it cannot intersect the line of P_1P_2, and, in particular, it cannot be on the segment P_1P_2.

```
so consider Z such that between Y Z Y' ∧ on_polypath [P1, P2, P3] Z
        from ... by (9.1)                                              24
obviously (by_ncols ∘ conjuncts) hence ¬on_polypath [P2, P3] Z from ...
    by (9.1), (I, 1), (II, 1)
hence on_polypath [P1, P2] Z from 24 by (9.1)                          25
have on_half_plane hp Y' from 12, 14, 23 by (7.8)
hence on_half_plane hp Z from ..., 24 by (7.9)                         26
hence between P1 Z P2 from ..., 25 by (9.1), (7.6)
qed from ..., 26 by (I, 2), (II, 1), (7.6)
```

Figure 11.12: Verification extract for the convex case of theorem 11.5

The verification shown in Figure 11.12 matches this argument's structure. We start by assuming there is some point Z on YY' which intersects the polygonal path $[P_1, P_2, P_3]$,

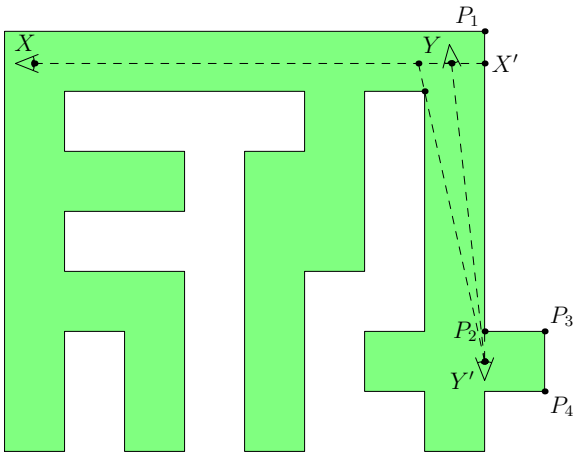


Figure 11.13: Edge-to-edge in a locally concave polygon

and proceed by contradiction (we have removed some of the references to earlier steps and inserted ellipses for readability).

11.4.2 Locally Concave Edges

When the edges are (strictly) locally concave, we have it that our starting point X is on the opposite side of the line P_1P_2 as the point P_3. In this case, we will generally need to "round the corner" $P_1P_2P_3$ to get line-of-sight with the next edge.

Before we apply Theorem 11.4, we will pick our destination. This will be the point Y' shown in Figure 11.13. This point is "just off" the vertex P_2, and can be found by ray-casting (Theorem 11.1) from P_2 in the direction $\overrightarrow{P_1P_2}$. We then use Theorem 11.4 as described in §11.3.3.1, moving along our initial line-of-sight XX' to obtain a new line-of-sight with Y'. The segment XYY' will then be the required path.

Now since Y' was found by ray-casting, we know it has line-of-sight to P_2. So all we need to do is rotate this line-of-sight to point at P_2P_3, which we know we can do using Theorem 11.4 as described in §11.3.3.2. We thus have the required path and line-of-sight to Y'' as shown in Figure 11.14.

We just have to confirm that the path XYY' does not intersect the polygon, and that $Y'Y''$ is a genuine line-of-sight. We know immediately that XY is off the polygon, since it is part of our original line-of-sight. This leaves us having to show that YY' and $Y'Y''$ do not intersect the polygon.

The way these segments were obtained via Theorem 11.4 guarantees that they do not intersect the fragment $P_3P_4 \ldots P_n$, so that leaves us to consider whether either of the

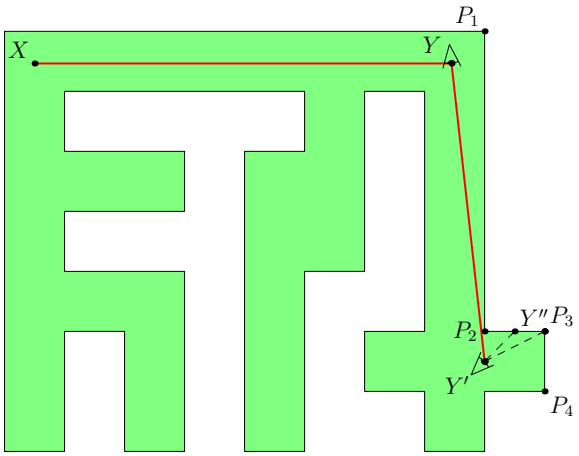


Figure 11.14: Edge-to-edge in a locally concave polygon (continued)

segments YY' and $Y'Y''$ intersect the fragment $P_1P_2P_3$.

It might seem that these cases would boil down to reasoning with half-planes again, since YY' and $Y'Y''$ are part of two rays emanating in opposite directions from the line of P_1P_2. But we were surprised to find that, when we set our discoverer at the problem, it showed that $Y'Y''$ is off the fragment $P_1P_2P_3$ and thus a line-of-sight according to incidence reasoning alone.

The discoverer also showed that YY' does not intersect the edge P_1P_2. We only need to use half-planes to show that it does not intersect the edge P_2P_3, using the same sort of tidy declarative verification that we saw in the last subsection: since Y lies between X and P_1P_2, and X and P_3 lie on opposite sides of P_1P_2, it must be the case that Y and P_3 lie on opposite sides of this line as well. And then, since Y' is on the line of P_1P_2, it follows from Theorem 7.8 that every point on YY' lies on the opposite side to P_3, and thus, does not lie on the line of P_1P_2.

This almost completes the verification. We have not talked about the case that P_1, P_2 and P_3 are collinear, which requires one application of Theorem 11.4, but we hope by this stage the reader is convinced we can achieve this, and is further convinced of how powerful our squeeze theorem is in allowing us to move between edges of a maze in a geometrically intuitive way.

11.4.3 Putting it all Together

We have packaged all the details discussed so far in this section into a single declarative verification consisting of 119 steps. That we can manage such long verifications shows

how easily Mizar Light, with our additional automated tools, can scale.

We would like to draw special attention to the hypotheses in the verified theorem (11.5). Note that we do not need to assume that we are dealing with simple polygons. We just have to assume that P_1P_2 does not intersect the rest of the path $P_2P_3 \dots P_n$, and that P_2P_3 does not intersect the rest of the path $P_3P_4 \dots P_n$. These are the minimal hypotheses we need to apply Theorem 11.4 as described in this section. We can think of them as saying that the polygon appears *locally* simple.

$$
\begin{aligned}
&\vdash \texttt{between}\ P_1\ X'\ P_2 \wedge P_2 \neq P_3 \\
&\quad \wedge \neg \texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ X \wedge \neg \texttt{on\_polypath}\ (P_3 :: Ps)\ P_2 \\
&\quad \wedge \neg (\exists Z.\ \texttt{between}\ X\ Z\ X' \wedge \texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ Z) \\
&\quad \wedge \neg (\exists Z.\ \texttt{between}\ P_1\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 :: P_3 :: Ps)\ Z) \\
&\quad \wedge \neg (\exists Z.\ \texttt{between}\ P_2\ Z\ P_3 \wedge \texttt{on\_polypath}\ (P_3 :: Ps)\ Z) \\
&\quad \Longrightarrow \exists Y.\ \exists Y'.\ \texttt{polypath\_connected}\ (\texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps))\ X\ Y \\
&\qquad \wedge \texttt{between}\ P_2\ Y'\ P_3 \wedge \neg \texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ Y) \\
&\qquad \wedge \neg \exists Z.\ \texttt{between}\ Y\ Z\ Y' \wedge (P_1 :: P_2 :: P_3 :: Ps)\ Z).
\end{aligned}
\tag{11.5}
$$

11.5 Without-Loss-of-Generality

In the last section, we described how to move edge-to-edge in a polygon. However, we effectively made a without-loss-of-generality assumption when stating the theorem, since we assumed that the edges in question were defined by the first three elements of the vertex list. More generally, they could be any two edges defined by three adjacent elements of the list, or, if we are considering moving forward from the last edge of a polygon back to the first, then the vertices defining the edges come from the last two elements and the first two elements.

We can follow Harrison's approach to dealing with without-loss-of-generality assumptions [37] if we can find an equivalence relation which preserves the properties of interest to us. In effect, we say that before we move a player to the next edge, we must be able to transform the polygon's vertex list into the player's perspective, much as we would perform a coordinate transform.

11.5.1 Polygon Rotations: Formulation

To make the idea work formally, we will need a way to represent a polygon rotation. Since our polygons are being represented by vertex lists, we must briefly leave the pleasant world of synthetic geometry and instead look at computing with lists. On the upside, some of these ideas are a general contribution to the theory of lists, and can be applied in other contexts.

It is not uncommon, for instance, to need to rotate a list. This can be understood in terms of splitting the list at some point and then switching the two halves. Formally:

$$\texttt{rotation\_of} : [\alpha] \to [\alpha] \to \texttt{bool}$$
$$\texttt{rotation\_of}\ ps\ qs \iff \exists xs.\ \exists ys.\ ps = xs + ys \wedge qs = ys + xs.$$

This defines an equivalence relation, but our use-case for list rotations gives us a complication. We represent a polygon by a vertex list where we assume that the first and last elements are duplicated. We cannot simply rotate this vertex list, since this will generally give us duplicate vertices and endpoints which do not match. Instead, we should only rotate the largest non-trivial prefix of the list, and then duplicate the new head at the end.

The definition of this sort of rotation is slightly more complex, but assuming the rotation is not the identity, we can understand it as taking some vertex inside the list and swapping it with the first and last elements. The sublists in between are then exchanged. In other words:

$$\begin{aligned}\vdash_{def} \texttt{rotation\_of}\ Ps\ Qs \iff& \\ \exists P.\ \exists Q.\ \exists Ps.\ \exists Qs.\ Ps &= ([P] + Ps' + [Q] + Ps'' + [P]) \\ \wedge\ Qs &= ([Q] + Ps'' + [P] + Ps' + [Q]) \\ \vee\ Ps &= Qs.\end{aligned}$$

We verified that this is an equivalence relation.

11.5.2 Invariance

In order to use these list rotations to justify without-loss-of-generality, we need to know that a rotation preserves the properties of interest. In our case, we need to know that a rotated polygon is still the same figure as a set of points, and we need to know that it is still *simple* as a set of segments.

The proofs are not exactly straightforward, since the set of points of a polygon and the simplicity of a polygon are defined in terms of list functions such as `adjacent`, `mem` and `pairwise`. The reasoning involves the interplay with these functions, for which we have had to contribute many new lemmas. We found it despairing to be doing this at such a late stage, when so much of the earlier verification was focused on synthetic declarative geometry. But with perseverance, we obtained the necessary theorem:

$$\begin{aligned} &\vdash \texttt{rotation\_of}\ Ps\ Qs \\ &\qquad \Longrightarrow (\texttt{simple\_polygon}\ Ps \Longrightarrow \texttt{simple\_polygon}\ Qs) \\ &\qquad\qquad \wedge\, \texttt{on\_polypath}\ Ps = \texttt{on\_polypath}\ Qs. \end{aligned} \tag{11.6}$$

11.5.3 Example

We end this section by explaining some of the reasoning that goes into using polygon rotations in the context of the main proof for this chapter. Suppose we have a polygon Ps as a vertex list, and we know we have line-of-sight to a point X' on an edge of the polygon. We can formally describe the position of X' with:

$$\exists P_i.\ \exists P_{i+1}.\ \texttt{mem}\ (P_i, P_{i+1})\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ P_i\ X'\ P_{i+1}.$$

The presence of `adjacent` is troublesome, but we have verified a theorem to help us out here:

$$\vdash \texttt{mem}\ (x,y)\ (\texttt{adjacent}\ xs) \iff \exists ws.\ \exists zs.\ xs = ws + [x,y] + zs.$$

This gives us just about everything we need to perform a rotation. We know that if the witnessed ys is empty, then our edge $[x,y]$ must already be at the front of the list. Otherwise, we know that the list xs is of the form

$$[w] + ws + [x,y] + zs + [w]$$

which is a rotation of

$$[x,y] + zs + [w] + ws + [x].$$

This puts the edge at the front of the list, as we require to apply Theorem 11.5. Our invariance theorem (11.6) assures us that the rotation will not change the set of points defined by the vertex list, nor affect the simplicity of the polygon.

11.6 Moving to Any Edge

We are now able to move between any two edges of a polygon. This takes us a significant way towards a verification of this last half of the Polygonal Jordan Curve Theorem.

In particular, we have shown that, given a point X with line-of-sight to a point X' on some edge, there must be a polygonal path-connected point Y which has line-of-sight to a point Y' on the *first* edge. Formally, we verify:

$$\begin{aligned}
\vdash\ & \texttt{simple_polygon}\ Ps \wedge \neg \texttt{on_polypath}\ Ps\ X \\
& \wedge \texttt{mem}\ (P, Q)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ P\ X'\ Q \\
& \wedge \neg(\exists Z.\ \texttt{between}\ X\ Z\ X' \wedge \texttt{on_polypath}\ Ps\ Z) \\
& \quad \Longrightarrow\ \exists Y.\ \exists Y'.\ \texttt{polypath_connected}\ (\texttt{on_polypath}\ Ps)\ X\ Y \\
& \qquad\qquad \wedge \neg \texttt{on_polypath}\ Ps\ Y \\
& \qquad\qquad \wedge \texttt{between}\ (\texttt{head}\ Ps)\ Y'\ (\texttt{head}\ (\texttt{tail}\ Ps)) \\
& \qquad\qquad \wedge \neg \exists Z.\ \texttt{between}\ Y\ Z\ Y' \wedge \texttt{on_polypath}\ Ps\ Z.
\end{aligned}$$

Note that for the very first time we are working on the hypothesis that our vertex list Ps defines a simple polygon. We can see why this hypothesis is needed by considering two of the hypotheses from Theorem 11.5, which we have said assumes that a polygon is only "locally" simple:

1. $\neg \texttt{on\_polypath}\ (P_3 :: Ps)\ P_2$;
2. $\neg(\exists Z.\ \texttt{between}\ P_1\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 :: P_3 :: Ps)\ Z)$.

By allowing Ps to be an arbitrarily rotated polygon, the first of these assumptions will require that all vertices are distinct and that no vertex appears on another edge. The second will require that the edges themselves do not intersect. This is just to say that the polygonal segment Ps must be simple.

To conclude this section, we will give some perspective on what is involved in the proof so far. In Figure 11.15, we have reproduced the example from §11.1, showing three players navigating a maze in order to have line-of-sight to the edge P_1P_2. However, we can now explain the peculiar shape of the red and blue paths, understanding that they have been obtained by precise applications of Theorem 11.4 using the auxiliary dashed lines. Each dashed line marks a point on a player's line-of-sight that they move

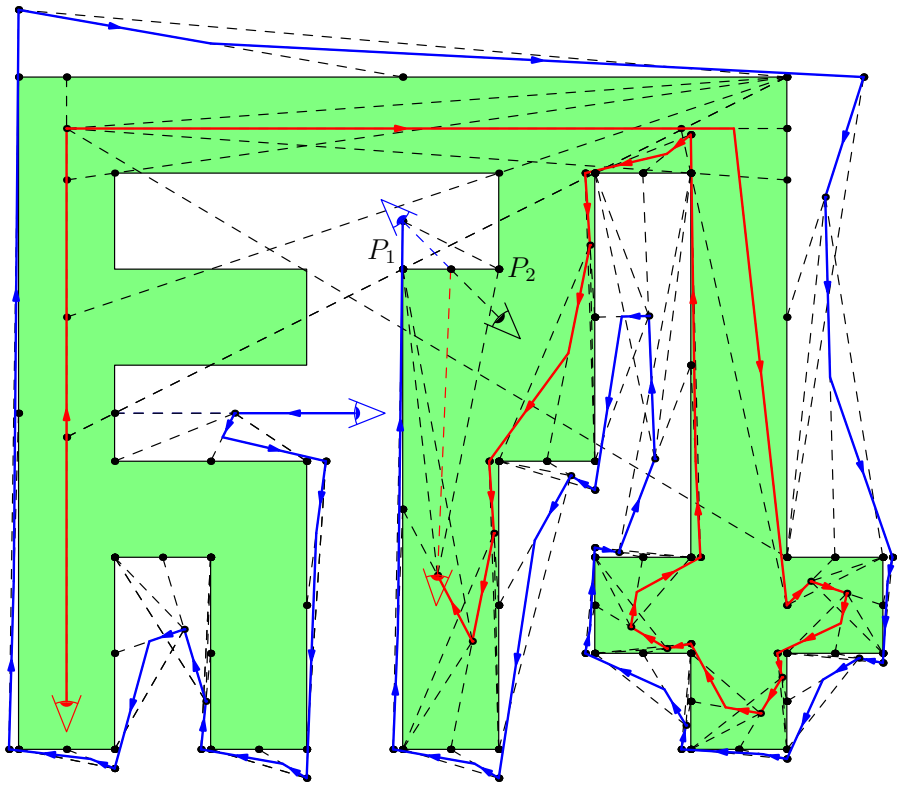


Figure 11.15: Navigating a maze with Theorem 11.4

towards, or a point on an edge towards which they rotate. Thus, we see each player navigating without compass or ruler, using only incidence and ordering.

11.7 Final Steps

We are almost there. To finish the proof, we will need to show that when two points are on the same side and looking at the same edge, then they are polygonal path-connected. We shall deal with this matter in §11.7.2. First, we shall deal with the neglected matter of how we are able to obtain line-of-sight to an edge in the first place. Both problems will have us reusing our near ubiquitous squeeze theorem (11.4).

11.7.1 Getting onto the Maze

If we take an arbitrary point X in the plane and an arbitrary point X' on a maze, then a simple ray-cast gives us line-of-sight to some other point X'' of the maze. If this point X'' lies between two other vertices, we have a line-of-sight to an edge.

However, if X'' coincides with a vertex as shown in Figure 11.16, we will need to

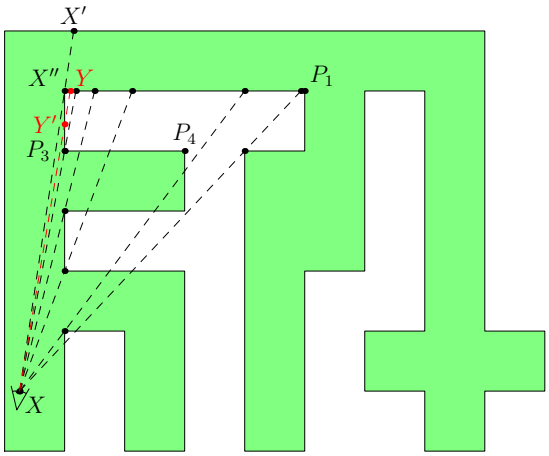


Figure 11.16: Obtaining line-of-sight with an edge

rotate our line-of-sight slightly. We will first assume, without loss-of-generality (or more accurately, we will assume we have rotated the polygon's vertex list) so that the vertex X'' coincides with the vertex P_2. We will want to obtain line-of-sight to either the edge P_1P_2 or P_2P_3.

We can do this by applying Theorem 11.4 to the edge P_1P_2 and the polygonal fragment $P_3P_4 \ldots P_n$ as described in §11.3.3.2. This will not generally give us our required point of intersection. In fact, it might be that there is *no* line-of-sight from X to the edge P_1P_2, because the edge P_2P_3 is in the way.

This does not pose much of a problem. If the segment XY intersects P_3X'' at the point Y', then we shall just take Y' as our line-of-sight. We just need two steps for this, one exploiting our linear reasoning tactic and the other our incidence discoverer, to show that the segment XY' is our required line-of-sight. On the other hand, if XY does not intersect P_3X'', then one step with our incidence discoverer verifies that it is the required line-of-sight.

We have formalised the result in Figure 11.17, retaining the without-loss-of-generality assumption. This allows us to review which hypotheses are actually needed for this particular theorem, and is one of the great benefits of formal verification. The labour involved in teasing out the necessary hypotheses means we usually end up with very tight lemmas with which to easily trace dependencies. Consider that this theorem appears in our theory file before simple polygons are even *defined*.

We can now get any point to have line-of-sight to the maze, and thus we can connect every point in the plane to another point with line-of-sight to any edge. This means that, if we have three points in the plane, we can find three paths which bring them

$$
\begin{aligned}
\vdash \neg\texttt{between}\ P_1\ P_3\ P_2 \wedge \neg\texttt{between}\ P_2\ P_1\ P_3 \wedge P_1 \neq P_2 \wedge P_1 \neq P_3 \\
\wedge \neg\texttt{on\_polypath}\ (P_3 :: Ps)\ P_2 \\
\wedge \neg(\exists Z.\ \texttt{between}\ P_1\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 :: P_3 :: Ps)\ Z) \\
\wedge \neg(\exists Z.\ \texttt{between}\ P_2\ Z\ P_3 \wedge \texttt{on\_polypath}\ (P_3 :: Ps)\ Z) \\
\wedge \neg\texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ X \\
\wedge \neg(\exists Z.\ \texttt{between}\ P_2\ Z\ X \wedge \texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ Z) \\
\Longrightarrow \exists Y.\ (\texttt{between}\ P_1\ Y\ P_2 \vee \texttt{between}\ P_2\ Y\ P_3) \\
\wedge \neg(\exists Z.\ \texttt{between}\ X\ Z\ Y \wedge \texttt{on\_polypath}\ (P_1 :: P_2 :: P_3 :: Ps)\ Z).
\end{aligned}
\tag{11.7}
$$

Figure 11.17: Rotating a line-of-sight to an edge

together at the same edge. It is enough now to verify that two of the three points are polygonal path-connected.

11.7.2 There are at Most Two Regions

Here is where we are: we have three points with line-of-sight to an edge P_1P_2 (without loss of generality). We know that two of these points X and Y are on the same side of P_1P_2. We will show that these two points can be polygonal path-connected.

First off, we have to consider that X and Y have line-of-sight to *different* points X' and Y'. Our diagrams so far in this chapter have given a different impression, but we must remember that the points obtained in our proofs ultimately rely on Axiom II, 2. This axiom allows us to extend a line-segment, but there are many possible witnesses for its existential conclusion, and we have generally allowed the abstraction to proliferate in our theory, rather than eliminating it with the epsilon-operator.

Consider the scenario depicted in Figure 11.18(a). Here, our points X and Y have line-of-sight to different points on the same edge, so our first order of business is to get X and Y to have line-of-sight to the same point. Again, we use Theorem 11.4 against the fragment $P_2P_3 \ldots P_n$, moving X along its line-of-sight to a point X'' so that it now has line-of-sight with Y'. A second application of the same theorem, moving along our new line-of-sight sees us facing the point Y. With the two points in each other's sights, we have the last piece of our path.

We are almost home free. We still need to factor in the fact that X and Y are on the same

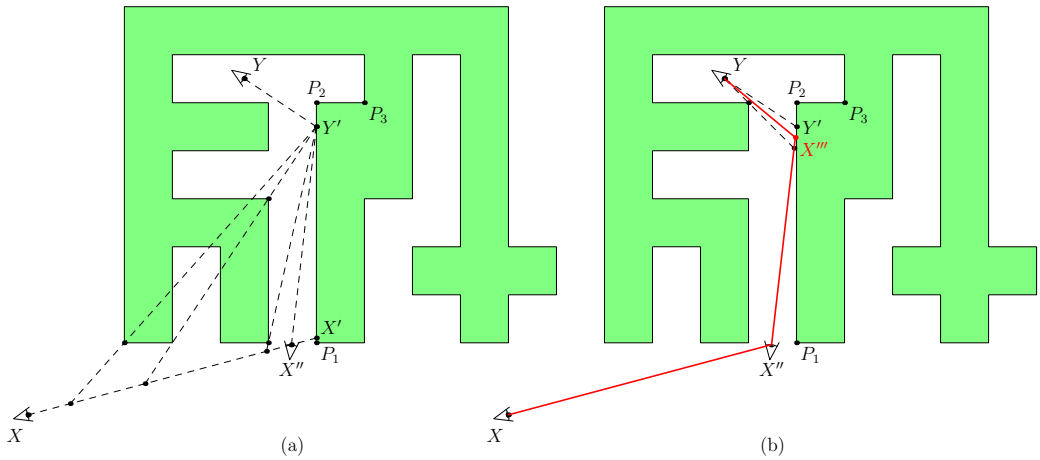


Figure 11.18: Connecting the final points

side of P_1P_2. This is needed because we only applied Theorem 11.4 to the fragment $P_2P_3 \ldots P_n$. The resulting lines-of-sight will not intersect this fragment, but we will need to account for the segment P_1P_2 separately.

Yet again, this is settled with our theorems for half-planes. Since X lies on the same side of P_1P_2 as Y, and X' lies on P_1P_2, all points on the segment XX'' must also lie on the same side. The same argument applies to the segment $X''Y'$, and finally to $Y'Y$. Since all of these points are on the same side of P_1P_2, they must lie off the line P_1P_2.

The final extract of the verified proof, witnessing the final path and showing this line of argument, is reproduced in Figure 11.19. We have omitted references to earlier steps.

$$
\begin{aligned}
\vdash\ & \neg \texttt{on\_polypath}\ (P_1 :: P_2 :: Ps)\ X \wedge \neg \texttt{on\_polypath}\ (P_1 :: P_2 :: Ps)\ Y \\
& \wedge \texttt{between}\ P_1\ X'\ P_2 \wedge \texttt{between}\ P_1\ Y'\ P_2 \\
& \wedge \neg(\exists Z.\ \texttt{between}\ X\ Z\ X' \wedge \texttt{on\_polypath}\ (P_1 :: P_2 :: Ps)\ Z) \\
& \wedge \neg(\exists Z.\ \texttt{between}\ Y\ Z\ Y' \wedge \texttt{on\_polypath}\ (P_1 :: P_2 :: Ps)\ Z) \\
& \wedge \neg(\exists Z.\ \texttt{between}\ P_1\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 :: Ps)\ Z) \\
& \wedge \texttt{on\_line}\ P1\ (\texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{on\_line}\ P2\ (\texttt{line\_of\_half\_plane}\ hp) \\
& \wedge \texttt{on\_half\_plane}\ hp\ X \wedge \texttt{on\_half\_plane}\ hp\ Y \\
& \Longrightarrow \texttt{polypath\_connected}\ (\texttt{on\_polypath}\ (P_1 :: P_2 :: Ps))\ X\ Y.
\end{aligned}
\tag{11.8}
$$

take $[X, X'', X''', Y]$

have on_line X' (line_of_half_plane hp)

$\wedge$ on_line Y' (line_of_half_plane hp)

from ... by (I, 2), (II, 1)

hence on_half_plane $hp\ X'' \wedge$ on_half_plane $hp\ X'''$

$\wedge \forall Z$. between $X''\ Z\ X''' \vee$ between $X'''\ Z\ Y$ from... by (7.8), (7.9)

have $\forall Z$. on_half_plane $hp\ Z \implies \neg$on_polypath $[P_1, P_2]\ Z$ from $3, 8$

by (9.1), (I, 2), (II, 1), (7.6)

hence $\forall Z$. on_polypath $[X'', X'', Y]\ Z \implies \neg$on_polypath $[P_1, P_2]\ Z$

from... by (9.1) 21

have $\forall Z$. between $X''\ Z\ X''' \implies \neg$on_polypath $(P_2 :: Ps)\ Z$

proof: fix Z

assume between $X''\ Z\ X'''$

hence between $X''\ Z\ Y'$ from ... using ORDER_TAC $\{X'', X''', Y', Z\}$

qed from ...

qed from ..., 21 by (9.1), (II, 1)

Figure 11.19: Verification Extract for Theorem 11.8

11.8 Conclusion

The proof of the final result, Theorem 9.4, puts all of the pieces together. We start from three arbitrary points in the plane not on the polygonal segment, have each obtain a line-of-sight to an edge of the maze, and then use polygon rotations and Theorem 11.5 to find three paths to three points with line-of-sight to the first edge of the maze. We then apply Theorem 7.4 which says that two of these points must be on the same side of the first edge, and thus, using Theorem 11.8, we complete the polygonal path which connects them.

As with the final theorem in the last chapter, the final theorem here is much more readable than the lemmas considered up to now. Those lemmas often have numerous complicated hypotheses, such as those for Theorem 11.8 (and remember that, for

brevity, we have omitted all their planar hypotheses). In setting them up as our intermediate goals, and going to the effort of verifying them, we must pay close attention in the hope that they will eventually entail our ultimate goal, a matter which is only guaranteed at the very end:

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ \alpha\ Ps \\
&\quad \wedge \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha \wedge \texttt{on\_plane}\ R\ \alpha \\
&\quad \wedge \neg \texttt{on\_polypath}\ Ps\ P \wedge \neg \texttt{on\_polypath}\ Ps\ Q \wedge \neg \texttt{on\_polypath}\ Ps\ R \\
&\quad \Longrightarrow \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q \\
&\qquad\quad \vee \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ R \\
&\qquad\quad \vee \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ Q\ R
\end{aligned}
\tag{9.4}
$$

Chapter 12

Conclusion

In an article published in the *American Mathematical Monthly*, on the validity of the Polygonal Jordan Curve Theorem in ordered geometry, almost forgotten, Guggenheimer laments:

> "It is astonishing that none of the textbooks of elementary axiomatic geometry gives a proof."
>
> [28, p. 753]

We find it more astonishing considering that the axioms of ordered geometry mark the first major contribution to the modern rigorisation of geometry by Pasch, "the father of rigour in geometry". It stretches all our credibility when those same axioms were delineated in the most influential textbook on modern axiomatic geometry, nearly eight decades earlier, and the theorem stated as derivable, by David Hilbert, one of the greatest mathematicians who ever lived.

Such is hindsight. We confess that we had little interest in the theorem initially. Hilbert implied it was trivial, that, in the *Grundlagen*, the interesting proofs only appear after the later groups of axioms. But he left literal labyrinths unexplored in the wake of those first two groups, and we would have passed them by if not for formal verification. It would have been easy to gloss over the details, to be convinced by invalid proofs — Veblen's, our own, and whatever Hilbert had in mind when he declared the result obtainable without much difficulty. Had that happened, we would not have done justice to ordered geometry.

But the details we must attend to in formal verification are, frankly, excessive. With current technology, we cannot simply translate even highly rigorous prose steps to for-

mal logic and expect generic automated tools to fill in the missing inferences, certainly not in a timely fashion.

This problem was compounded for us, we conjecture, because we chose a domain which was synthetic, not algebraic. Here, there is little room for the equational reasoning that characterises algebraic proofs, and where powerful term rewriting systems can be made the workhorse of automation. Instead, we have implicational theorems with complex antecedents, and too many variables which need explicit instantiation.

Happily, as we saw in Chapter 5, it turned out that the domain was amenable to a tailored proof-search. Since the inferences needed to fill the proof gaps almost never introduce points into a geometric configuration, we can exhaustively search for all the incidence relations that hold in a figure. Restricting our attention to a small finite geometry in this way, characterised by sets of collinear, non-collinear and planar points, and the rules between them, we were faced with a tractable combinatorial space.

It is possible that there are other theories for which we can carve out such a space, and which have so far resisted formalisation for lack of ways to tailor a search of it. We have shown, by following the example of traditional theorem proving languages [70], how a combinator language could be used to rapidly prototype the appropriate search strategies using a very modest amount of declarative code. As an additional benefit, the strategies, themselves built by composition rules, are naturally extensible and composable across domains. They could potentially scale well to ever more complex problems.

A crucial assumption of our search language is that the data found (such as sequents about which sets of points are collinear) grows cumulatively. Later data are assumed not to replace earlier data, and no data are discarded as new information comes in. These are not generally realistic assumptions, and already posed some inefficiences for us when we came to think about point equalities. Such data should ideally force all prior data to be normalised.

Nevertheless, the tool was highly effective at dealing with incidence arguments, and with it, we typically found that our new verifications corresponded one-to-one with the prose. That is, our verifications were structured just as if the formal steps were direct translations of steps in the prose argument. This ideal is rarely achieved in declarative proof, and gives us new hope that formal verification can one day play a major role in ordinary mathematics. It is as if we had managed to recreate the silent mechanical reasoning capacity of an artificial Hilbert, allowing us to write a much more *human*

verification without worrying about slacking on rigour.

It was crucial to have this automation by the time we came to verify the Polygonal Jordan Curve Theorem. Typically, those working in formal verification transcribe proofs that they know to be basically correct. We, on the other hand, were trying to patch a proof that we were convinced was wrong. We only dared the venture because we were confident that our incidence automation made it feasible. Otherwise, we could scarcely imagine how we would have completed it.

The verification consisted of roughly 1400 declarative proof steps. Of these, 213 are simple `assume` steps that come with no justification. Another 215 are `consider` steps which introduce geometric entities. 116 are `fix` steps for naming the quantified variables. Of the remaining 849 intermediate steps, 111 were assisted by the incidence automation of Chapter 4 and 82 by the combination of incidence automation and linear reasoning automation of Chapter 6. Going by the results of Chapter 5, we may have replaced over 500 proof steps about tedious incidence reasoning alone.

These are steps that had been difficult to obtain by hand. Some we had missed entirely, something we could expect to happen frequently on the substantially larger verification of the Polygonal Jordan Curve Theorem, leading to mental blocks which may then have sent us down spurious paths, or worse, had us give up entirely. But with the incidence details taken care of, our proofs could focus on the less combinatorial steps, those that interest a model geometer, and which can be sketched out on paper and subjected to the full resources of intuitive observation.

One lesson we would take from the need and efficacy of our automation is that the current ideal of declarative verification, where the user only has to think in terms of lemmas and their dependencies, letting only *generic* automation connect the dots, is probably going to be overly optimistic for some time to come. Domain specific automation, crafted by the user, will still be needed, and those working in formal verification must continue to think as computer scientists and programmers. Theorem provers must continue to cater to their needs, as HOL Light does, putting its users directly at the interpreter of a powerful programming language so that they can craft new proof tools highly tailored to the problem at hand.

We would like to conclude by reflecting a little on our experience verifying the Polygonal Jordan Curve Theorem. We believe that formal verification holds a unique place in the general space of human problem solving. The chief point is that, in a verification, the goal is stated up front without any ambiguity, and the circumstances under

which that goal is achieved are likewise unambiguous. The result is authoritative and not subject to any human review.

We had the problem stably formalised very early on, and so the prospect of its solution was almost tangible. We just needed the theorem prover to produce the formalised problem statement as a fully fledged theorem. However, we did not anticipate the effort required to get there. This was not because the verification was a slog, for our automation took care of the pedantic mechanical details. It was just that the problem seemed to grow in complexity as we tackled it, and the theorem prover could not give us a reliable measure of our distance from the goal.

We could sweat out a major lemma such as 10.17 from Appendix C, but with theorems such as these, which have half a dozen opaque hypotheses and are based on admittedly convoluted definitions, we found it difficult to convince ourselves that the theorems said exactly what we wanted them to say and that they put us on the right track. At any time, we were prepared for the possibility that we were navigating ourselves into a dead-end, having misunderstood some crucial aspect of the problem. We struggled to find our bearings and aim a true course, and the effort required made it difficult to communicate any sense of real progress to others. We were quite alone in this maze.

So when the theorem prover finally and curtly announced “No subgoals”, it was like turning yet another dull corner and then suddenly emerging disoriented in bright daylight. There was no need to trace back over our tracks, scrutinising and carefully reassuring ourselves that we had indeed found the exit. In an instant, we knew that we had solved the problem that had started it all, that our strategy was vindicated and all previous uncertainties had evaporated.

In practised mathematics, a proof must convince human readers. The subtle complexities of its solution must be teased out and magnified lest they mislead us into fallacy. But a verification is wholly unlike this. Already convinced that our verifier is sound, and having been long convinced that the relatively short problem statement is the one we intend, the verification is complete the moment the computer declares it so. Looking back over the verification, we can start accepting the subtleties with some suspension of disbelief, and not feel obliged to investigate every last nook and cranny: the machine has vouched for us. The path of definitions and lemmas can instead be left as a story offering only insight on the complexities involved, along with some nuggets of useful techniques and strategies for other adventurers.

Our normal scintillas of doubt reflect the diligence due in rigorous problem solving, arising from a sense of personal responsibility for our errors and a desire to be honest and not mislead readers. Such doubts would rightly beg us to revisit and elaborate our proofs from time-to-time. With verification, they are only misplaced. Even the most rigorous of axiomatic geometers must envy the resulting sense of a task so definitively complete.

Bibliography

[1] John R. Anderson. Obituary: Herbert A. Simon (1916–2001). *American Psychologist*, 56:516–518, 2001.

[2] Y. Balashov and M. Janssen. Presentism and Relativity. *The British Journal for the Philosophy of Science*, 54(2):327–346, 2003.

[3] Friedrich L. Bauer. Calculations Using Symbols. In *Origins and Foundations of Computing*, pages 23–40. Springer Berlin Heidelberg, 2010.

[4] Garrett Birkhoff and Mary Bennett. Hilbert's *Grundlagen der Geometrie*. *Rendiconti del Circolo Matematico di Palermo*, 36:343–389, 1987.

[5] Richard Boulton. *Efficiency in a Fully-Expansive Theorem Prover*. PhD thesis, Cambridge University, 1993.

[6] Carl B. Boyer. *A History of Mathematics*. John Wiley & Sons, 1991.

[7] Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques D. Fleuriot, editors, *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 89–109. Springer, 2013.

[8] B. Buchberger. A Theoretical Basis for the Reduction of Polynomials to Canonical Forms. *ACM SIGSAM Bulletin*, 10(3):19–29, 1976.

[9] HOL Light CHANGELOG. `http://code.google.com/p/hol-light/source/browse/trunk/CHANGES`.

[10] Ze-wei Chen. Efficient Access to Knowledge via Forward Chaining Tactics. Technical report, Cornell University, April 1995.

[11] Alonzo Church. A Set of Postulates for the Foundation of Logic. *Annals of Mathematics*, 33(2):pp. 346–366, 1932.

[12] Alonzo Church. A Formulation of the Simple Theory of Types. *Association for Symbolic Logic*, 5:56–68, 1940.

[13] Simon Colton, Alan Bundy, and Toby Walsh. On The Notion Of Interestingness In Automated Mathematical Discovery. *International Journal of Human-Computer Studies*, 53(3):351–375, 2000.

[14] Simon Colton and Stephen Muggleton. ILP for Mathematical Discovery. In Tams Horvth and Akihiro Yamamoto, editors, *Inductive Logic Programming*, volume 2835 of *Lecture Notes in Computer Science*, pages 93–111. Springer Berlin Heidelberg, 2003.

[15] Haskell B. Curry. The Inconsistency of Certain Formal Logic. *The Journal of Symbolic Logic*, 7(3):pp. 115–117, 1942.

[16] Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, Principles of Programming Languages, pages 207–212, New York, NY, USA, 1982. ACM.

[17] James H. Davenport and Joos Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(12):29 – 35, 1988.

[18] Brian Davies. Whither Mathematics? *Notices of the American Mathematical Society*, 52(11), 2005.

[19] N.G. de Bruijn. The Mathematical Vernacular, a language for Mathematics with typed sets. In P. Dybjer et al, editor, *Proceedings from the Workshop on Programming Logic*, volume 37, 1987.

[20] Nicolaas Govert de Bruijn. The mathematical language AUTOMATH, its usage, and some of its extensions. In *Symposium on automatic demonstration*, pages 29–61. Springer, 1970.

[21] Christophe Dehlinger, Jean-François Dufourd, and Pascal Schreck. Higher-Order Intuitionistic Formalization and Proofs in Hilbert's Elementary Geometry. In *Automated Deduction in Geometry: Revised Papers from the Third International Workshop on Automated Deduction in Geometry*, volume 2061, pages 306–324, London, UK, 2001. Springer-Verlag.

[22] Jean-François Dufourd. Discrete Jordan Curve Theorem: A proof formalized in Coq with hypermaps. In *25th International Symposium on Theoretical Aspects of Computer Science*, pages 253–264, 2008.

[23] Solomon Feferman. Mathematical Intuition Vs. Mathematical Monsters. *Synthese*, 125(3):317–332, 2000.

[24] Georg Feigl. Über die elementaren Anordnungssätz. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 33, 1925.

[25] Georges Gonthier. The Four Colour Theorem: Engineering of a Formal Proof. In Deepak Kapur, editor, *Computer Mathematics*, volume 5081 of *Lecture Notes in Computer Science*, page 333. Springer, 2007.

[26] Georges Gonthier, Andrea Asperti, Jeremy Avigad, Yves Bertot, Cyril Cohen, Franois Garillot, Stphane Le Roux, Assia Mahboubi, Russell OConnor, Sidi Ould Biha, Ioana Pasca, Laurence Rideau, Alexey Solovyev, Enrico Tassi, and Laurent Thry. A machine-checked proof of the odd order theorem. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving*, volume 7998 of *Lecture Notes in Computer Science*, pages 163–179. Springer Berlin Heidelberg, 2013.

[27] H. Guggenheimer. The Jordan Curve Theorem and an Unpublished Manuscript by Max Dehn. *Archive for History of Exact Sciences*, 17:193–200, 1977.

[28] Heinrich W Guggenheimer. The Jordan and Schoefiles Theorems in Axiomatic Geometry. *The American Mathematical Monthly*, 85(9):pp. 753–756, 1978.

[29] Thomas Hales. Introduction to the Flyspeck Project. `http://drops.dagstuhl.de/opus/\newlinevolltexte/2006/432/pdf/05021.HalesThomas.Paper.432.pdf`.

[30] Thomas Hales. Formal proof. *Notices of the American Mathematical Society*, 55:1370–1380, 2008.

[31] Thomas C. Hales. Jordan's Proof of the Jordan Curve Theorem. *Studies in Logic, Grammar and Rhetoric*, 10(23), 2007.

[32] Thomas C. Hales. The Jordan Curve Theorem, Formally and Informally. *The American Mathematical Monthly*, 114:882894, 2007.

[33] John Harrison. A Mizar Mode for HOL. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *Theorem Proving in Higher Order Logics*, volume 1125 of *Lecture Notes in Computer Science*, pages 203–220, Turku, Finland, 1996. Springer-Verlag.

[34] John Harrison. Formalized Mathematics. Technical Report 36, Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, 1996.

[35] John Harrison. HOL Light: a Tutorial Introduction. In *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*, volume 1166, pages 265–269. Springer-Verlag, 1996.

[36] John Harrison. Towards self-verification of hol light. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the third International Joint Conference, IJCAR 2006*, volume 4130 of *Lecture Notes in Computer Science*, pages 177–191, Seattle, WA, 2006. Springer-Verlag.

[37] John Harrison. Without Loss of Generality. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proceedings of the 22nd International Conference on Theorem Proving in Higher Order Logics, 2009*, volume 5674 of *Lecture Notes in Computer Science*, pages 43–59, Munich, Germany, 2009. Springer-Verlag.

[38] Thomas L. Heath. *Euclid: The Thirteen Books of The Elements*, volume 2. Dover Publications, 1956.

[39] Henri Poincaré. Poincaré's Review of Hilbert's *Foundations of Geometry*. *Bulletin of the American Mathematical Society*, 10(1):1–23, 1903.

[40] Henri Poincaré. *Science and Method*. Cosimo Classics, 2007.

[41] David Hilbert. *The Foundations of Geometry*. The Open Court Publishing Company, 1st edition, 1950.

[42] David Hilbert. *Foundations of Geometry*. Open Court Classics, 2nd edition, 1971. Translated from the 10th edition of the *Grundlagen der Geometrie*.

[43] David Hilbert. On the Infinite. In Paul Benacerraf and Hilary Putnam, editors, *Philosophy of Mathematics: Selected Readings*. Cambridge University Press, 1984.

[44] Peter V. Homeier. The HOL-Omega Logic. In *Theorem Proving in Higher Order Logics*, pages 244–259, 2009.

[45] Edward V. Huntington and J. Robert Kline. Sets of Independent Postulates for Betweenness. *Transactions of the American Mathematical Society*, 18(3):pp. 301–325, 1917.

[46] Xiao-Shan Gao Jing-Zhong Zhang, Shang-Ching Chou. Automated Production of Traditional Proofs for Theorems in Euclidean geometry. *Annals of Mathematics and Artificial Intelligence*, 13(1-2):109–138, 1995.

[47] Moa Johansson, Lucas Dixon, and Alan Bundy. Conjecture Synthesis for Inductive Theories. *Journal of Automated Reasoning*, 47(3):251–289, 2011.

[48] Dale M. Johnson. Prelude to Dimension Theory: The Geometrical Investigations of Bernard Bolzano. *Archive for History of Exact Sciences*, 17:261–295, 1977. 10.1007/BF00499625.

[49] Jordan, Camille. *Cours d'analyse de l'École polytechnique*. Gauthier-Villars et fils Paris, 1893.

[50] Florian Kammuller and Markus Wenzel. Locales: A Sectioning Concept for Isabelle. In *Theorem Proving in Higher Order Logics*, volume 1690, pages 149–165. Springer, 1999.

[51] Vladimir Kanovei and Michael Reeken. A nonstandard proof of the Jordan curve theorem. *Pacific Journal of Mathematics*, 36(1):219–229, 1971.

[52] Susumu Katayama. Recent Improvements to MagicHaskeller. In Ute Schmid, Emanuel Kitzelmann, and Rinus Plasmeijer, editors, *Approaches and Applications of Inductive Programming*, volume 5812 of *Lecture Notes in Computer Science*. Springer, 2009.

[53] H. C. Kennedy. Origins of Modern Axiomatics: Pasch to Peano. *The American Mathematical Monthly*, 79:133–136, 1972.

[54] Oleg Kiselyov, Chung chieh Shan, Daniel P. Friedman, and Amr Sabry. Backtracking, Interleaving, and Terminating Monad Transformers. In *International Conference on Functional Programming*, pages 192–203, 2005.

[55] Wilbur Richard Knorr. *The Evolution of the Euclidean Elements*. Springer, February 1974.

[56] Ramana Kumar and Tjark Weber. Validating QBF Validity in HOL4. In *Interactive Theorem Proving*, pages 168–183, 2011.

[57] Ondej Kunar. Proving Valid Quantified Boolean Formulas in HOL Light. In Marko Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving*, volume 6898 of *Lecture Notes in Computer Science*, pages 184–199. Springer Berlin Heidelberg, 2011.

[58] Edmund Landau. *Foundations of Analysis*. Chelsea Pub Co, 2001.

[59] N. J. Lennes. Theorems on the Simple Finite Polygon and Polyhedron. *American Journal of Mathematics*, 33(1):pp. 37–62, 1911.

[60] Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing Desargues' theorem in Coq using ranks. In *Symposium on Applied Computing*, pages 1110–1115, 2009.

[61] Robert Vaughn Main. *A Critique of the Incidence and Order Axioms of Geometry*. PhD thesis, Oregon State University, 1970.

[62] Conor McBride and Ross Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(1):1–13, 2008.

[63] Roy L. McCasland and Alan Bundy. MATHsAiD: A Mathematical Theorem Discovery Tool. In *SYNASC*, pages 17–22, 2006.

[64] Robert McNaughton. Review: Alfred Tarski, A Decision Method for Elementary Algebra and Geometry. *Bulletin of the American Mathematical Society*, 59(1):91–93, 1953.

[65] Michael A. McRobbie and John K. Slaney, editors. *Optimizing proof search in model elimination*, volume 1104 of *Lecture Notes in Computer Science*. Springer, 1996.

[66] Laura I. Meikle and Jacques D. Fleuriot. Formalizing Hilbert's Grundlagen in Isabelle/Isar. In *Theorem Proving in Higher Order Logics*, volume 2758, pages 319–334. Springer, 2003.

[67] Elliot Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 4th edition, 1997.

[68] Jia Meng, Claire Quigley, and Lawrence C. Paulson. Automation for interactive proof: First prototype. *Inf. Comput.*, 204(10):1575–1596, 2006.

[69] Christopher P Wadsworth Michael J Gordon, Arthur J Milner. *Edinburgh LCF*. Springer-Verlag, 1979.

[70] R. Milner and R. S. Bird. The Use of Machines to Assist in Rigorous Proof [and Discussion]. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 312(1522):pp. 411–422, 1984.

[71] Robin Milner, Mads Tofte, Robert Harper, and David Macqueen. *The Definition of Standard ML - Revised*. The MIT Press, rev sub edition, May 1997.

[72] Omar Montano-Rivas, Roy McCasland, Lucas Dixon, and Alan Bundy. Scheme-Based Synthesis of Inductive Theories. In Grigori Sidorov, Arturo Hernndez Aguirre, and CarlosAlberto Reyes Garca, editors, *Advances in Artificial Intelligence*, volume 6437 of *Lecture Notes in Computer Science*, pages 348–361. Springer Berlin Heidelberg, 2010.

[73] Omar Montano-Rivas, Roy McCasland, Lucas Dixon, and Alan Bundy. Scheme-based theorem discovery and concept invention. *Expert Systems with Applications*, 39(2):1637 – 1646, 2012.

[74] Robert Lee Moore. On a Set of Postulates which Suffice to Define a Number-Plane. *Transactions of the American Mathematical Society*, 16(1):pp. 27–32, 1915.

[75] Alan Musgrave. Logicism revisited. *The British Journal for the Philosophy of Science*, 28(2):pp. 99–127, 1977.

[76] Julien Narboux. A Decision Procedure for Geometry in Coq. In *Theorem Proving in Higher Order Logics*, volume 3223, pages 225–240. Springer, 2004.

[77] Julien Narboux. Mechanical Theorem Proving in Tarski's Geometry. In *Automated Deduction in Geometry*, volume 4869, pages 139–156, 2006.

[78] Victor Pambuccian. Forms of the Pasch axiom in ordered geometry. *Mathematical Logic Quarterly*, 56(1):29–34, 2010.

[79] Victor Pambuccian. The axiomatics of ordered geometry: I. Ordered incidence spaces. *Expositiones Mathematicae*, 29(1):24 – 66, 2011.

[80] Art Quaife. Automated Development of Tarski's Geometry. *Journal of Automated Reasoning*, 5(1):97–118, 1989.

[81] Richard Courant and Herbert Robbins. *"What is Mathematics?"*. Oxford University Press, 1996.

[82] Bertrand Russell. Mathematical Logic as Based on the Theory of Types. *American Journal of Mathematics*, 30(3):pp. 222–262, 1908.

[83] Russell, Bertrand. *The Principles of Mathematics*. Merchant Books, 2008.

[84] Jeremy Yallop Sam Lindley and Phil Wadler. Idioms are oblivious, arrows are meticulous, monads are promiscuous. In *Mathematically Structured Functional Programming*, 2008. no proceedings.

[85] Vesna Pavlovic Sana Stojanovic and Predrag Janicic. A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs. In *Automated Deduction in Geometry*, Lecture Notes in Computer Science, pages 201–220. Springer, 2010.

[86] Phil Scott. Mechanising Hilbert's *Foundations of Geometry* in Isabelle. Master's thesis, University of Edinburgh, 2008.

[87] Phil Scott and Jacques D. Fleuriot. An Investigation of Hilbert's Implicit Reasoning through Proof Discovery in Idle-Time. In *Automated Deduction in Geometry*, Lecture Notes in Computer Science, pages 182–200. Springer, 2010.

[88] Phil Scott and Jacques D. Fleuriot. Composable Discovery Engines for Interactive Theorem Proving. In *Interactive Theorem Proving*, volume 6898 of *Lecture Notes in Computer Science*, pages 370–375. Springer, 2011.

[89] Phil Scott and Jacques D. Fleuriot. A Combinator Language for Theorem Discovery. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics - 11th International Conference*, volume 7362 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2012.

[90] Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. D. Reidel Publishing Company, 1988.

[91] Shang-Ching Chou, Xiao-Shan Gao, Jing-Zhong Zhang. *Machine Proofs in Geometry*. World Scientific Publishing, 1994.

[92] J. Michael Spivey. Algebras for combinatorial search. *Journal of Functional Programming*, 19(3-4):469–487, 2009.

[93] Michael Spivey. Combinators for Breadth-First Search. *Journal of Functional Programming*, 10(4):397–408, 2000.

[94] S. Doaitse Swierstra, Pablo R. Azero Alcocer, and João Saraiva. Designing and Implementing Combinator Languages. In *Advanced Functional Programming*, volume 1608 of *Lecture Notes in Computer Science*, pages 150–206. Springer, 1999.

[95] Alfred Tarski and Steven Givant. Tarski's System of Geometry. *Bulletin of Symbolic Logic*, 5:175–214, 1999.

[96] Wen tsün Wu. *Mechanical Theorem Proving in Geometries*. Springer-Verlag Wien New York, 1994.

[97] Helge Tverberg. A Proof of the Jordan Curve Theorem. *Bulletin of the London Mathematical Society*, 12:34–38, 1980.

[98] L. S. van Benthem Jutting. *Checking Landau's Grundlagen in the Automath system*. PhD thesis, Eindhoven University of Technology, 1977.

[99] Oswald Veblen. Hilbert's Foundations of Geometry. *The Monist*, 13(2):pp. 303–309, 1903.

[100] Oswald Veblen. A System of Axioms for Geometry. *Transactions of the American Mathematical Society*, 5:343–384, 1904.

[101] Oswald Veblen. Theory on Plane Curves in Non-Metrical Analysis Situs. *Transactions of the American Mathematical Society*, 6(1):83–98, 1905.

[102] Philip Wadler. Monads for Functional Programming. In *Advanced Functional Programming*, pages 24–52, 1995.

[103] Hermann Weyl. David Hilbert and his mathematical work. *Bulletin of the American Mathematical Society*, 50:635, 1944.

[104] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume 1 of *Principia mathematica*. Cambridge University Press, 1927.

[105] Freek Wiedijk. Mizar Light for HOL Light. In *Theorem Proving in Higher Order Logics*, volume 2152, pages 378–394, London, UK, 2001. Springer-Verlag.

[106] Freek Wiedijk, editor. *Introduction*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[107] Freek Wiedijk. Mizar's Soft Type System. In *Theorem Proving in Higher Order Logics*, volume 4732, pages 383–399. Springer, 2007.

[108] Sean Wilson and Jacques Fleuriot. Geometry Explorer: Combining Dynamic Geometry, Automated Geometry Theorem Proving and Diagrammatic Proofs. In *Proceedings of UITP 2005 (User Interfaces for Theorem Provers)*, Apr 2005.

[109] L. Wittgenstein. *Tractatus Logico-philosophicus*. International Library of Psychology, Philosophy, and Scientific Method. Harcourt, Brace, Incorporated, 1922.

[110] Edward Z Yang. Adventures in Three Monads. *The Monad Reader*, page 11, 2010.

[111] Pierre Castéran Yves Bertot. *Interactive Theorem Proving and Program Development*. Springer, 2004.

Appendix A

Elementary Consequences of Group II

$$\begin{aligned}\vdash\ &\neg\texttt{collinear}\ \{A,B,C\} \wedge \neg\texttt{collinear}\ \{A,D,E\} \wedge \neg\texttt{collinear}\ \{C,D,E\} \\ &\wedge \texttt{planar}\ \{A,B,C,D,E\} \wedge \texttt{between}\ A\ D\ B \\ &\Longrightarrow \exists F.\ \texttt{collinear}\ \{D,E,F\} \wedge (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C)\end{aligned} \tag{3.11}$$

$$\begin{aligned}\vdash\ &\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\wedge \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a) \\ &\wedge \neg(\exists a.\ \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a) \\ &\wedge (\exists \alpha.\ \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \\ &\qquad \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha) \\ &\wedge \texttt{between}\ A\ D\ B \\ &\Longrightarrow \exists F.\ (\exists a.\ \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a) \\ &\qquad (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C)\end{aligned} \tag{A.1}$$

$$\begin{aligned}&\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ C \\ &\qquad \Longrightarrow \exists F.\ \texttt{between}\ D\ E\ F \wedge \texttt{between}\ A\ F\ B\end{aligned} \tag{5.2}$$

$$\begin{aligned}&\neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ B \\ &\qquad \Longrightarrow \exists F.\ \texttt{between}\ D\ F\ E \wedge \texttt{between}\ A\ F\ C\end{aligned} \tag{5.3}$$

$$\vdash A \neq C \Longrightarrow \exists D.\ \texttt{between}\ A\ D\ C \qquad \text{(THEOREM 3)}$$

$$\begin{aligned}&\vdash \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a\\ &\quad \wedge\ A \neq B \wedge A \neq C \wedge B \neq C \qquad \text{(THEOREM 4)}\\ &\quad \Longrightarrow \texttt{between}\ A\ B\ C \vee \texttt{between}\ B\ A\ C \vee \texttt{between}\ A\ C\ B\end{aligned}$$

$$\begin{aligned}&\vdash (\texttt{between}\ A\ B\ C \wedge \texttt{between}\ B\ C\ D \Longrightarrow \texttt{between}\ A\ B\ D \wedge \texttt{between}\ A\ C\ D)\\ &\quad \wedge (\texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \Longrightarrow \texttt{between}\ A\ B\ D \wedge \texttt{between}\ B\ C\ D)\\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(THEOREM 5)}\end{aligned}$$

$$A \neq B \Longrightarrow \texttt{infinite}\ \{P | \texttt{between}\ A\ P\ B\}$$

$$\begin{aligned}&\vdash \neg\texttt{on\_line}\ A\ a \wedge \neg\texttt{on\_line}\ B\ a \wedge \neg\texttt{on\_line}\ C\ a\\ &\quad \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a \qquad (7.3)\\ &\quad \Longrightarrow \neg\texttt{between}\ A\ D\ B \vee \neg\texttt{between}\ A\ E\ C \vee \neg\texttt{between}\ B\ F\ C\end{aligned}$$

A.1 Half-Planes

$$\vdash \exists P.\ \texttt{on\_half\_plane}\ hp\ P$$

$$\begin{aligned}&\vdash \texttt{on\_half\_plane}\ hp\ P \wedge \texttt{on\_half\_plane}\ hq\ P\\ &\quad \wedge \texttt{line\_of\_half\_plane}\ hp = \texttt{line\_of\_half\_plane}\ hq\\ &\quad \Longrightarrow hp = hq\end{aligned}$$

$$\begin{aligned}&\vdash \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad \Longrightarrow \exists! hp.\ \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hp)\\ &\qquad\qquad \wedge \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hp)\\ &\qquad\qquad \wedge \texttt{on\_half\_plane}\ hp\ C\end{aligned}$$

$$\begin{aligned}\vdash &(\forall P.\ \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha)\\ &\implies \exists hp.\ \exists hq.\ hp \neq hq\\ &\quad \wedge a = \texttt{line\_of\_half\_plane}\ hp \wedge a = \texttt{line\_of\_half\_plane}\ hq\\ &\quad \wedge (\forall P.\ \texttt{on\_plane}\ P\ \alpha\\ &\qquad \iff \texttt{on\_line}\ P\ a \vee \texttt{on\_half\_plane}\ hp\ P \vee \texttt{on\_half\_plane}\ hq\ P)\end{aligned} \tag{7.4}$$

$$\begin{aligned}\vdash &(\forall R.\ \texttt{on\_half\_plane}\ hp\ R \implies \texttt{on\_plane}\ R\ \alpha) \wedge \texttt{on\_half\_plane}\ hp\ P\\ &\implies (\texttt{on\_half\_plane}\ hp\ Q\\ &\quad \iff \neg(\exists R.\ \texttt{on\_line}\ R\ (\texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{between}\ P\ R\ Q)\\ &\qquad \wedge \texttt{on\_plane}\ Q\ \alpha \wedge \neg\texttt{on\_line}\ Q\ (\texttt{line\_of\_half\_plane hp}))\end{aligned} \tag{7.7}$$

$$\begin{aligned}\vdash &\texttt{on\_line}\ P\ (\texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{on\_half\_plane}\ hp\ Q\\ &\implies \texttt{between}\ P\ Q\ R \vee \texttt{between}\ P\ R\ Q \implies \texttt{on\_half\_plane}\ hp\end{aligned} \tag{7.8}$$

$$\begin{aligned}\vdash &\texttt{on\_half\_plane}\ hp\ P \wedge \texttt{on\_half\_plane}\ hp\ R\\ &\implies \texttt{between}\ P\ Q\ R \implies \texttt{on\_half\_plane}\ hp\ Q\end{aligned} \tag{7.9}$$

A.2 Rays

$$\vdash \exists P.\ \texttt{on\_ray}\ r\ P$$

$$\vdash \texttt{on\_ray}\ r\ P \wedge \texttt{on\_ray}\ s\ P \wedge \texttt{ray\_origin}\ r = \texttt{ray\_origin}\ s \implies r = s$$

$$\begin{aligned}\vdash &\texttt{on\_ray}\ r\ P \wedge \texttt{on\_line}\ P\ a \wedge \texttt{on\_line}\ (\texttt{ray\_origin}\ r)\ a\\ &\implies \texttt{on\_ray}\ r\ Q \implies \texttt{on\_line}\ Q\ a\end{aligned}$$

$$\vdash \neg\texttt{on\_ray}\ r\ (\texttt{ray\_origin}\ r)$$

$$\vdash \texttt{on\_line}\ P\ a$$
$$\Longrightarrow\ \exists r.\ \exists s.\ r \neq s \wedge P = \texttt{ray\_origin}\ r \wedge Q = \texttt{ray\_origin}\ s$$
$$\wedge\ (\forall X.\ \texttt{on\_line}\ X\ a \iff X = \texttt{ray\_origin}\ r \vee \texttt{on\_ray}\ r\ X \vee \texttt{on\_ray}\ s\ X)$$

$$\vdash (\forall R.\ \texttt{on\_ray}\ r\ R \implies \texttt{on\_line}\ R\ a) \wedge \texttt{on\_ray}\ r\ P$$
$$\Longrightarrow\ (\texttt{on\_ray}\ r\ Q$$
$$\iff\ Q \neq \texttt{ray\_origin}\ r \wedge \neg\texttt{between}\ P\ (\texttt{ray\_origin}\ r)\ Q \wedge \texttt{on\_line}\ Q\ a$$

Appendix B

Polygonal Jordan Curve Theorem: Full Specification

B.1 HOL Light List and Set Library

Function specifications marked with † have been contributed.

$$\texttt{head} : [\alpha] \to \alpha \qquad \vdash_{def} \texttt{head}\ (x :: xs) = x$$

$$\texttt{tail} : [\alpha] \to [\alpha]\dagger \qquad \vdash_{def} \texttt{tail}\ [] = [] \qquad \vdash_{def} \texttt{tail}\ (x :: xs) = xs$$

$$\texttt{length}\ [\alpha] \to \mathbb{N} \qquad \vdash_{def} \texttt{length}\ [] = 0 \qquad \vdash_{def} \texttt{length}\ (x :: xs) = \texttt{length}\ xs + 1$$

$$\texttt{butlast} : [\alpha] \to [\alpha] \qquad \vdash_{def} \texttt{butlast}\ [] = [] \qquad \vdash_{def} \texttt{butlast}\ (x :: xs) = \texttt{if}\ xs = []\ \texttt{then}\ []\ \texttt{else}\ x :: (\texttt{butlast}\ xs)$$

$$\texttt{el} : \texttt{int} \to [\alpha] \to \alpha \qquad \vdash_{def} \texttt{el}\ 0\ xs = \texttt{head}\ xs \qquad \vdash_{def} \texttt{el}\ (\texttt{suc}\ n)\ xs = \texttt{el}\ n\ (\texttt{tail}\ xs)$$

$\texttt{mem} : \alpha \to [\alpha] \to \texttt{bool}$
$\vdash_{def}\ \texttt{mem}\ x\ [] = \bot$
$\vdash_{def}\ \texttt{mem}\ x\ (y :: ys) = x = y \vee \texttt{mem}\ x\ ys$

$\texttt{all} : (\alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool}$
$\vdash_{def}\ \texttt{all}\ p\ [] = \bot$
$\vdash_{def}\ \texttt{all}\ p\ (x :: xs) = p\ x \wedge \texttt{all}\ p\ xs$

$\texttt{pairwise} : (\alpha \to \alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool}$
$\vdash_{def}\ \texttt{pairwise}\ R\ [] = \top$
$\vdash_{def}\ \texttt{pairwise}\ R\ (x :: xs) = \texttt{all}\ (Rx)\ xs \wedge \texttt{pairwise}\ R\ xs$

$\texttt{zip} : [\alpha] \to [\beta] \to [(\alpha, \beta)]$
$\vdash_{def}\ \texttt{zip}\ []\ [] = []$
$\vdash_{def}\ \texttt{zip}\ (x :: xs)(y :: ys) = (x, y) :: \texttt{zip}\ xs\ ys$

$\texttt{adjacent} : [\alpha] \to [(\alpha, \alpha)]\dagger$
$\vdash_{def}\ \texttt{adjacent}\ xs = \texttt{zip}\ (\texttt{butlast}\ xs)\ (\texttt{tail}\ xs)$

$\texttt{disjoint} : (\alpha \to \texttt{bool}) \to (\alpha \to \texttt{bool}) \to \texttt{bool}$
$\vdash_{def}\ \texttt{disjoint}\ S\ T = S \cap T = \emptyset$

B.2 Polygon Definitions

$\texttt{on\_polypath} : [\texttt{point}] \to \texttt{point} \to \texttt{bool}$
$\vdash_{def}\ \texttt{on\_polypath}\ Ps\ P \iff$
$\quad \texttt{mem}\ P\ Ps \vee \exists x.\ \exists y.\ \texttt{mem}\ (x, y)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ x\ P\ y.$

$\texttt{polypath\_connected} : \texttt{plane} \to (\texttt{point} \to \texttt{bool}) \to \texttt{point} \to \texttt{point} \to \texttt{bool}$
$\vdash_{def}\ \texttt{polypath\_connected}\ \alpha\ figure\ P\ Q \iff$
$\quad \exists path.\ path \neq []$
$\qquad \wedge\ (\forall R.\ \texttt{mem}\ R\ path \implies \texttt{on\_plane}\ R\ \alpha)$
$\qquad \wedge\ \texttt{head}\ path = P \wedge \texttt{last}\ path = Q$
$\qquad \wedge\ \texttt{disjoint}\ (\texttt{on\_polypath}\ path)\ figure.$

$$\texttt{simple\_polygon} : \texttt{plane} \to [\texttt{point}] \to \texttt{bool}$$

$$\vdash_{def} \texttt{simple\_polygon}\ \alpha\ Ps \iff$$
$$3 \leq \texttt{length}\ Ps$$
$$\wedge\, \texttt{head}\ ps = \texttt{last}\ Ps$$
$$\wedge\, (\forall P.\ \texttt{mem}\ P\ Ps \implies \texttt{on\_plane}\ P\ \alpha)$$
$$\wedge\, \texttt{pairwise}\ (\neq)\ (\texttt{butlast}\ Ps)$$
$$\wedge\, \neg(\exists P.\ \exists Q.\ existsX.\ \texttt{mem}\ X\ Ps \wedge \texttt{mem}\ (P,Q)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ P\ X\ Q)$$
$$\wedge\, \texttt{pairwise}\ (\lambda(P,Q)\ (P',Q').$$
$$\neg(\exists X.\texttt{between}\ P\ X\ P' \wedge \texttt{between}\ Q\ X\ Q')\ (\texttt{adjacent}\ Ps)).$$

B.3 Theorems

$$\vdash \texttt{on\_plane}\ P1\ \alpha \wedge \texttt{on\_plane}\ P2\ \alpha \wedge \texttt{on\_plane}\ Q1\ \alpha \wedge \texttt{on\_plane}\ Q2\ \alpha$$
$$\wedge\, (\forall X.\ \texttt{mem}\ X\ Ps \implies \texttt{on\_plane}\ X\ \alpha) \wedge (\forall X.\texttt{mem}\ X\ Qs \implies \texttt{on\_plane}\ X\ \alpha)$$
$$\wedge\, \neg(\exists a.\ \texttt{on\_line}\ P1\ a \wedge \texttt{on\_line}\ P2\ a \wedge \texttt{on\_line}\ Q1\ a)$$
$$\wedge\, \neg(\exists a.\ \texttt{on\_line}\ Q1\ a \wedge \texttt{on\_line}\ Q2\ a \wedge \texttt{on\_line}\ P1\ a)$$
$$\wedge\, \texttt{between}\ P1\ X\ P2 \wedge \texttt{between}\ Q1\ X\ Q2$$
$$\wedge\, P1 = \texttt{last}\ (P2 :: Ps) \wedge Q1 = \texttt{last}\ (Q2 :: Qs)$$
$$\implies \exists Y.\texttt{on\_polypath}\ (P2 :: Ps\ Y) \wedge \texttt{on\_polypath}\ (Q1 :: Q2 :: Qs)\ Y$$
$$\vee\, \texttt{on\_polypath}\ (P1 :: P2 :: Ps)\ Y) \wedge \texttt{on\_polypath}\ (Q2 :: Qs)\ Y$$

$$\vdash \texttt{simple\_polygon}\ \alpha\ Ps$$
$$\implies \exists P.\ \exists Q.\ \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha$$
$$\wedge\, \neg\texttt{on\_polypath}\ Ps\ P \wedge \neg\texttt{on\_polypath}\ Ps\ Q$$
$$\wedge\, \neg\texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q$$

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ \alpha\ Ps \\
&\quad \wedge \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha \wedge \texttt{on\_plane}\ R\ \alpha \\
&\quad \wedge \neg\texttt{on\_polypath}\ Ps\ P \wedge \neg\texttt{on\_polypath}\ Ps\ Q \wedge \neg\texttt{on\_polypath}\ Ps\ R \\
&\quad\ \Longrightarrow \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q \\
&\qquad\quad \vee \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ R \\
&\qquad\quad \vee \texttt{polypath\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ Q\ R
\end{aligned}
$$

Appendix C

Polygonal Jordan Curve Theorem: Supporting Theorems

$$\texttt{in\_triangle}\ (A,B,C)\ P \implies \neg(\exists a.\ \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \tag{C.1}$$

$$\begin{aligned}
&Qs = [P] + Ps + [P]\\
&\wedge\ \texttt{on_plane}\ A\ \alpha \wedge \texttt{on_plane}\ B\ \alpha \wedge \texttt{on_plane}\ C\ \alpha \wedge \texttt{on_plane}\ C'\ \alpha\\
&\wedge\ (\forall X.\ \texttt{mem}\ X\ Qs \implies \texttt{on_plane}\ X\ \alpha)\\
&\wedge\ \neg\texttt{on_polypath}\ Qs\ A\\
&\wedge\ (\texttt{between}\ A\ B\ B' \vee \texttt{between}\ A\ B'\ B \vee A \neq B \wedge B \neq B')\\
&\wedge\ \neg(\exists X.\ \texttt{on_polypath}\ [B,B']\ X \wedge \texttt{on_polypath}\ Qs\ X)\\
&\wedge\ \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B\ a \wedge \texttt{on_line}\ C\ a)\\
&\wedge\ \neg(\exists a.\ \texttt{on_line}\ A\ a \wedge \texttt{on_line}\ B'\ a \wedge \texttt{on_line}\ C'\ a)\\
&\implies \exists \Gamma'.\ \texttt{polyseg_crossings}\ (A,B,C)\ (\Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs))\\
&\qquad\qquad (\texttt{adjacent}\ Qs)\\
&\quad = \texttt{polyseg_crossings}\ (A,B',C')\ (\Gamma_{final}\ (A,B',C')\ \Gamma'\ (\texttt{adjacent}\ Qs))\\
&\qquad\qquad (\texttt{adjacent}\ Qs)
\end{aligned} \tag{10.17}$$