Ordered Geometry in Hilbert's *Foundations*

Phil Scott

Doctor of Philosophy

Centre for Intelligent Systems and their Applications

School of Informatics

University of Edinburgh

2013

Abstract

In this thesis, we initiate a formal analysis of David Hilbert's axiomatisation of ordered geometry as it appears in the tenth and current edition of his celebrated *Foundations of Geometry*. Using Hilbert as a case-study, we argue that with the help of an interactive proof assistant, we are able to critically analyse the structure of mathematical arguments with an unprecedented level of confidence, provided we stay faithful to the text itself. We use a readable declarative language to support this, verify proofs synthetically as much as possible, and embed automation to recover the computational detail that a mathematician would find tedious. The thesis ends in a large case study, verifying a proof of the Polygonal Jordan Curve Theorem in the very general setting of *ordered geometry*.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Phil Scott*)

Table of Contents

Chapter 1

System Description

In this chapter, we describe our chosen verification tool HOL Light and its object logic and we address a few philosophical questions about using computerised logical systems to analyse mathematical foundations. The chapter is provided to keep the thesis self-contained, and contains very little in the way of new material. To assist the reader wanting to skip topics familiar or otherwise irrelevant to them, we give a brief overview of each section.

In §1.1, we give an overview of simple type theory, including its history. In §1.1.1, we give its formal definition.

In §1.2, we review the LCF approach to implementing interactive proof assistants, and our chosen proof assistant HOL Light.

In §1.3.4, we introduce the idea of a declarative proof, and in §1.3.5, we discuss the Mizar Light language that we have used for our verification. In §1.3.6, we present previously unpublished material on some useful extensions and modifications to the Mizar Light language.

1.1 Object Logic

The logic of our proof assistant is Church's Simple Theory of Types [7] or the simply-typed lambda calculus. This calculus is typically associated with the foundations of programming languages, but in fact, it was always conceived as a foundation for logic [6], and a solution to the paradoxes which plagued näive set theories. Church regarded it as less "artificial" than ZF set theory and Russell's Ramified Type Theory [63].

The original lambda calculus turns out to be inconsistent (all of its terms are provably equivalent [9]), and ironically, Church fixed the problem in essentially the same way as Russell, though by this period in time, 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 mathematics which is nominally based on first-order set theory, but this is not the case. The *scope* of mathematics might well be characterised by reference to ZFC set theory, as evidenced by its power to close down research programmes such as the Continuum Problem in the face of Cohen's independence proof (EDIT: I've lost my citation to this. I think it's in a paper by Field). And it can be argued that the concerns of model theory has one favouring first-order logic [15]. But practised verification usually favours typed higher-order logics. Russell and Whitehead's celebrated computer unassisted verification [81] was in a typed higher-order logic. The first computer assisted non-trivial verification by De Bruijn in AUTOMATH used a powerful extension of the simply typed lambda calculus. And according to Wiedijk's survey [83], eleven of the world's seventeen proof assistants are based on a higher-order logic. Even those based nominally on untyped set theory such as Mizar [11] can be viewed as being typed [84].

We suggest the reason for this is down to the pervasive use of metatheory in mathematics, which requires more expressive power to verify than offered by first-order logic alone. The entirety of Chapter 5 is a case-study, based in Hilbert's *Foundations of Geometry*, on why faithful mathematical verification demands this expressive power.

The simple theory of types as it appears in HOL Light makes an extension to Church's simple theory of types: it turns the type-schemas which appear implicitly in Church's original paper into type variables, or *polymorphic types*. This extension is so standard that hereafter when we refer to "simple type theory" we shall mean one in which polymorphic types replace type-schemas. The extension does not increase the expressive power of the logic, nor does it add any syntactic burden: as with Standard ML [54], it is possible to mechanically infer the most general type of any term [10].

1.1.1 Definitions

We now give the formal definition of the object logic as used in HOL Light. We will be somewhat terse in this section, since we are just aiming to keep the thesis self-contained. For a more detailed introduction to this material, we recommend the excellent HOL Light manual [25].

1.1.1.1 Syntax

First of all, we introduce *types*, which act as disjoint 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 $\tau_1 \rightarrow \tau_2$ is a (function) type. Arrow composition is right-associative, so that $\alpha \rightarrow \beta \rightarrow \gamma$ parses as $\alpha \rightarrow (\beta \rightarrow \gamma)$.

Next, we have *terms* based on an alphabet of term constants and term variables, and a typing relation (:) which is used to associate a term with its type, according to the rules:

1. Every term constant and every term variable is a term.
2. For terms $f : \tau_1$ and $x : \tau_2$, we have that $f\ x$ is a term such that $f\ x : \tau_1 \rightarrow \tau_2$. These combinations are left-associative: $f\ g\ h = (f\ g)\ h$. This means that contrary to conventional mathematical notation, we write function and predicate applications as $f\ x$ rather than $f(x)$.
3. Given a term variable $x : \tau_1$ and a term $y : \tau_2$, we have that $\lambda x.y$ is a term such that $\lambda x.y : \tau_1 \rightarrow \tau_2$. These lambda abstractions consume everything to the right, so $\lambda x.f\ x\ y$ parses as $\lambda x.((f\ x)\ y)$.

It should be clear from these definitions that the set of possible terms is now constrained by types. So, for instance, according to these rules, the classic Russell paradox $\lambda x.\neg(x\,x)$ is not a term, since the x cannot be given a type consistent with rules 2 and 3.

The idea here is that an object of type $\alpha \rightarrow \beta$ is a function with domain α and codomain β. A lambda abstraction in this type is then a function literal, and the notation $\lambda x.f\ x$ can be understood in much the same way as the notation $x \mapsto f\ x$ which is perhaps more familiar to mathematicians.

This 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 the type of truth values. Predicates, functions and relations are also unified in a single syntax for higher-order functions. A predicate is now just a function sending objects of some type α to $\top$ when they satisfy the predicate, and to $\bot$ when they do not. These predicates can be used to represent sets, and we can recover simple set-theoretic operations such as union and intersection as functions from predicates to predicates.

All functions in simple type theory have arity 1. If one wants an n-ary function with $n > 1$, they either turn the single argument into an n-tuple, or they have the function map its first argument to another function which expects the remaining $n-1$ arguments. 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 $f\ x\ y$, rather than the more cumberson $(f\ x)\ y$.

1.1.1.2 Calculus

For inference rules, HOL Light needs to introduce a primitive type `bool` which is to be inhabited by truth values, and one primitive constant representing equality $(=)\ :\ \alpha \to \alpha \to \texttt{bool}$. With these, we can introduce the notion of *judgements* or *sequents*, which have the form $\{A_1 : \texttt{bool}, A_2, \ldots, A_n\} \vdash C\ :\ \texttt{bool}$. We are to understand these judgements as saying that C is judged true in the context of assumptions $\{A_1, A_2, \ldots, A_n\}$. These judgements are then linked by inference rules, which say how new judgements arise from existing judgements. In other words, we prove judgements, rather than propositions.

The rules of HOL Light's calculus (Figure 1.1) are few, and 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.

Finally, we have a few rules to handle instantiations of free variables and type variables. We have a rule to substitute free terms t_1, …, t_n with term variables x_1, …, x_n. We have another rule to substitute type variables α_1, …, α_n with types τ_1, …, τ_n.

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

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

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

$$\frac{\Gamma \vdash x = y \quad f = g}{\Gamma \cup \Delta \vdash f\,x = f\,y}\ \texttt{MK\_COMB} \qquad \text{for any } x,\ \frac{s = t}{(\lambda x.s) = \lambda x.t}\ \texttt{ABS}$$

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

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

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

Figure 1.1: Inference Rules

1.1.2 Higher-order Logic

These inference rules embed a full (intuitionistic) higher-order logic (hereafter *HOL*). For instance, it is possible to convince oneself that a conjunction $p \wedge q$ can be formulated as $(p,q) = (\top,\top)$ which can, in turn, be formulated as

$$(\lambda f.f\ p\ q) = \lambda f.f\ ((\lambda x.x) = \lambda x.x)\ ((\lambda x.x) = \lambda x.x).$$

Indeed, with a suitable formulation of implication $\implies$, we can verify the full specification for these two symbols:

$$p \wedge q \implies p \qquad p \wedge q \implies q \tag{1.1}$$

$$(r \implies p) \wedge (r \implies q) \implies r \implies p \wedge q. \tag{1.2}$$

All other connectives and quantifiers can similarly be embedded, and from here on, we will assume the following pieces of derived syntax, in order of precedence.

$\neg$	Negation
$\wedge$	Conjunction
$\vee$	Disjunction
$\implies$	Implication
$\iff$	Equivalence
$\forall$	Universal quantification
$\exists$	Existential quantification
$\texttt{let}\ x_1 = t_1, \ldots, x_n = t_n\ \ldots$	Local variable declaration
`if...then...else...`	Conditionals

Quantifiers are “greedy”, having a scope which binds everything to their right.

1.2 Proof Assistant

Our chosen proof assistant is HOL Light [23], though throughout this thesis we shall mention earlier work [65] that was carried through in Isabelle/HOL [60]. Both systems use the exact same object logic, and we developed them in such a way that it is reasonably straightforward to port our code between them.

We should say a brief word on notation. To make the thesis more accessible, we try, wherever possible, to use conventional mathematical notation rather than HOL Light syntax. Those wishing to see the syntactic translations we assume in our formalisation should consult Appendix E.

1.2.1 Edinburgh LCF

HOL Light is a theorem prover in the tradition of Edinburgh LCF [52], implemented in the Ocaml programming language. The roots of both go back to the original statically typed functional language ML.

If natural languages serve as the metalanguages for defining object logics for human proof checkers, then ML serves as the metalanguage for defining object logics for *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” in place of “inference rule”.

The use of a programming language can actually clarify a lot. Consider the definition of the `ABS` inference rule in Figure 1.1, where we say “for all x”, 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 metavariable x as an additional argument, while that for `BETA` does not.

While programming languages clarify, they might also be viewed suspiciously. We want our formal verifications to inspire a virtually indefeasible level of confidence, and as of writing, computers are justifiably untrustworthy. ML and HOL Light are notable by being part of a tradition which puts safety first. We define the data-type of sequents as *abstract*, thereby hiding the implementation and exposing only the term data-type and inference rules.

ML's strong type system enforces this boundary between a *trusted kernel* and the outside world. Because of this enforcement, it is impossible to construct a HOL sequent through any method other than correct derivation from inference rules, assuming we trust the implementation of the kernel.

In order to acquire trust in HOL Light's implementation, one can simply peruse its modest 672 line kernel. Many of the implementation details in this kernel are trivial. The only non-trivial parts which might recommend careful inspection are 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"). That the code implementing these behaviours is less trivial than the rest of the kernel implementation is reflected in the fact that, when defining a correct logical calculus in ordinary mathematics, one usually has to pay careful attention to the behaviour of free-substitutability.

1.2.2 Additional Functionality

HOL Light extends the simple theory of types by allowing us to add new term and type definitions, and new axioms. Definitions cannot contain free variables on their right hand sides, nor can they be recursive. Thus, it is trivial that they lead only to conservative extensions of the theory. We could potentially substitute the right hand sides of all our definitions throughout a theory without affecting the validity of the derivations.

HOL Light also extends the simple theory of types by allowing one to define an abstract type of values in bijection with a non-empty subset of an existing type. This feature is particularly useful for enforcing that derived definitions are used in a way which "respects" the abstraction.

For instance, in geometry, we might wish to define a "ray" as a particular kind of set of points, and we may wish to define an "endpoint" of a ray as a set of points with additional constraints (see Chapter 6). We will want to use expressions such as "endpoint of a ray", but we probably wish to declare expressions such as "endpoint of a line" or "endpoint of a plane" as meaningless. These expressions break the abstraction of "ray", but the abstraction can be reinforced by using a derived abstract type. As an added benefit, because HOL Light types can always be mechanically reconstructed from expressions, abstract types can keep our formal statements short by pushing the constraints on what defines a ray into an automatically inferred type.

1.3 Classical Logic

HOL Light as described so far is non-classical. The only primitive values we have considered are those of type `bool`, and contrary to classical logic, the calculus does not require that `bool` has only two inhabitants.

1.3.1 Choice and Extensionality

To make HOL classical, a new term is introduced, ε, and two new axioms:[1]

$$\forall P\,x.\ P\,x \implies P\ (\varepsilon x.P\,x) \qquad \forall t.(\lambda x.t\ x) = t$$

We are to understand $\varepsilon x.P$ as "the arbitrarily chosen x satisfying P". This construct is used both as a definitional tool, but it is also equivalent, with the other classical axioms, to the full axiom of choice. It should be contrasted with the weaker ι binder, which yields expressions $\iota x.P$ to be read as "the uniquely specified x satisfying P." The distinction is discussed further in §5.5.

The axiom governing this notion 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 some sense, well-defined, since from the perspective of the logic's model theory, all terms 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 $\varepsilon x.P\ x$.

More accurately, the only statements we can derive of $\varepsilon x.P\ x$ are logical truths. If we take Wittgenstein at his word in the *Tractacus*, 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 other axiom is the axiom of extensionality, or the η-reduction axiom. With it, one can prove that any two lambda expressions $\lambda x.f\ x$ and $\lambda x.g\ x$ are equal precisely when we can prove $f\ x = g\ x$ for arbitrary x.

It turns out that with these two axioms we can derive the law of excluded-middle [13], by exploiting the fact that ε terms with extensionally equivalent bodies must now pick out the same value. In particular, if we consider $\varepsilon x.x \vee P$ and $\varepsilon x.\neg P \vee P$, we realise that,

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

in assuming P, ε must pick the same value for these two expressions, since the bodies of the ε terms are now extensionally equivalent. With a little bit more work, one then derives the law of excluded-middle.

1.3.2 Axiom of Infinity

The final classical axiom in HOL Light is the axiom of infinity. We will discuss this axiom in some detail in Chapter 5. 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 from our verification. This is not intended to be a serious philosophical statement, but we might draw a parallel here between our *post-hoc* deletion of the axiom of infinity from higher-order logic with Hilbert's later expressed scepticism on the security of infinite sets [29].

1.3.3 Proof Tools

Outside of the relatively tiny HOL Light kernel is a huge collection of proof tools written in ML. We can think of these tools as programs which generate derivations, whose validity is guaranteed by the enforced boundary between the HOL Light kernel and the outside world. In many cases, we do not care how these tools work, or whether they even contain bugs: if they generate the sequent we want, encapsulation of the kernel tells us that their generated derivation was nevertheless correct.

1.3.3.1 Tactics

HOL Light implements the LCF tactic system [53], in which we understand the process of proving a theorem T as the solving of a goal by breaking it into subgoals. We will need to briefly discuss the implementation of tactics for §1.3.4, since we make use of some of the lower level details.

As far as tactics go, a *goal* is a pair consisting of a term T together with *hypotheses*. There is some confusing overloading here, since a sequent is similarly a pair, where we have a conclusion together with *assumptions*. In our experience, it is essential not to confuse the two, since in HOL Light, the hypotheses are actually *sequents*!

The purpose of a tactic is to input a goal and to produce 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, after which, a fully machine checked theorem can be recovered.

Many tactics correspond to inference rules but applied in reverse. For instance, there is a tactic `DISCH_TAC` which can be applied when one's goal is an implication $P \implies Q$. This deletes the current subgoal and replaces it with the goal Q but now under the *hypothesis* P.

1.3.3.2 Fully Automated Procedures

Some tactics in HOL Light are decision procedures or otherwise fully automated proof tools. Generally, these are used to solve a goal *outright*, without generating any new subgoals. HOL Light has decision procedures for checking tautologies, a decision procedure for linear arithmetic, and for computing ideal elements via Gröbner bases [4].

More recently, the tools have expanded outside of ML. Eekelen et al [41] have implemented HOL Light code to take proof certificates from external "off-the-shelf" checkers for universally quantified propositional logic. These are formally verified by being converted to derivations of HOL Light sequents (the verification is fully expansive [2]). Similar progress has been made in HOL4 [40] and Isabelle [51].

1.3.4 Declarative Proof

Proof assistants accept formal texts in broadly two types: declarative and procedural. The difference is analogous to that between declarative and procedural programming languages. In the procedural approach, the user employs tactics to compose automated tools in order to verify formalised theorems.

Different tactic languages have their own styles and idioms, but they usually support both *forward* reasoning from the premises of a theorem 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 procedural proof scripts.

Declarative proof assistants on the other hand, beginning with *Mizar* [11], have attempted to imitate the style of ordinary mathematics. The proof scripts express the flow of argument as trees of intermediate results, branching at subproofs, with each intermediate result stated with its dependencies. The flow of the script always moves *forward* from assumptions to goal, with the focus on *what* the logical relations between formulas are, rather than *how* the proof state is transformed to represent such relations. This latter detail is left to the internal operation of the proof assistant, which tries to use automated proof tools to bridge the inferential gaps.

Declarative style proof seemed natural to us when it came to formalising synthetic geometry, and we felt that so long as we were striving to analyse Hilbert's prose arguments, it made sense to try to produce proofs that matched the prose at least structurally. We would also suggest that procedural proofs, relying so often on contextual rewriting, do not work particularly well in the domain of synthetic geometry. This is a difficult claim to back up, because it might be that we could have fixed the "problem" with better representations. For now, we just note that the majority of our lemmas and theorems are non-equational formulas with large numbers of hypotheses. These are difficult to apply as simplification rules.

1.3.5 Mizar Light

Mizar Light, developed by Wiedijk [82], is a declarative style proof language embedded in HOL Light and inspired by the primitives of the declarative proof assistant, Mizar [11]. We give an overview of its primitives in Figure 1.2. 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 [24], the operational semantics of these proof steps can be given in terms of tactics and simplified goals. The tree of goal-stacks becomes the tree of subproofs. The hypotheses of each goal become the intermediate theorems one has deduced during the proof. The various syntactic primitives of the declarative proof language become tactics which drive the proof *forward*. Here, for instance, is a typical line one might read in one of our Mizar Light proof scripts:

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

This `consider` step translates to a tactic which introduces the subgoal `∃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 [47], but additional tactics can be composed with the `using` keyword. The justification tactic usually needs some help to solve its goals, and in declarative proof, we provide justification theorems using the keyword `by`. In this example, we have added some theorems (I, 2 and I, 3.1)

Primitive	Meaning
`theorem` *term*	Begins a proof of *term*.
proof *proof*	Asserts *proof* as a justification for the current step.
assume *term*	Asserts *term* as a justified assumption at this point.
so	Refers to the previous step as justifying the current step.
`have` *term*	Asserts *term* as derivable at this point.
thus *term*	Asserts *term* as derivable at which point the (sub)theorem is justified.
`hence` *term*	As `so thus` *term*
take *var*	Identifies *var* as the witness for the (sub)theorem.
fix *vars*	Establishes *vars* as fixed but arbitrary variables.
consider *vars* `st` *term*	Introduces *vars* witnessing *term*.
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 *term*	Syntactic sugar to identify the supposition of each *case*.
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 1.2: An overview of Mizar Light

which are passed directly to MESON.

These Mizar Light commands are ordinary ML functions, and the Mizar Light language is really just a combinator language [72]. This has been useful to us, since it is almost trivial to modify and extend Mizar Light by just 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 3 and 5). Another feature of tactics is that they compose algebraically. There are tensoring operators, sums, identities and a zero. In trying to keep to the spirit of HOL Light, we have designed our own automation in Chapter 3 in terms of an algebraic combinator language.

1.3.6 Declarative Interactivity

Wiedijk's basic combinators are based on the original Mizar system, a batch prover, and so his Mizar Light proof scripts are written in their entirety and then evaluated in one. We find this undesirable, firstly, because the error reporting is not rich enough to show where errors occur in the case of a failed proof. Secondly, we have chosen to implement our automation (described in Chapter 3) so that it runs concurrently as a user writes a proof. Here, the tool works best when it can exploit the user's idle time when working *interactively* as opposed to *batch* mode.

The problem here lies with case-splitting. Here are the original combinators at work in an extract of one of Wiedijk's example proofs (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]]]];
...
```

Notice firstly that this is a subproof within a larger proof. Notice secondly that it contains two nested subproofs, case-splitting on the propositions `p1 = p2` and `¬(p1 = p2)`. The steps of each subproof are collected in lists, which makes for a neatly structured proof document, where the proof tree is reflected by ML data-structures. However, the steps of an interactive proof are supposed to be applied *linearly*, one-by-one, traversing the implicit proof tree. Here is what we prefer to write at the top-level (> 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]
```

We could probably achieve this linearisation by rewriting Wiedijk's proof system in a continuation-passing style. However, the development of HOL Light has emphasised backward-compatibility, and so wherever possible, we wish to build on existing implementation.

1.3.6.1 Interactive Case-splits

Case splits are ultimately justified by proving a disjunction of all considered cases. However, in the Mizar-style of proof and as is common in ordinary mathematical proof, 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

$$p1 = p2 \lor \neg(p1 = p2)$$

does not appear in the proof text. Instead, it is assembled by the `per` combinator from the first element of each subproof. The step then folds a case-splitting tactic over the list of cases, incorporating the tactics generated by their respective subproofs.

The important point here is that the implicit disjunction must be determined before any of the subproof tactics are 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 [24]. In his system, the disjunction is assembled at the very end of the proof, when all goals have been solved and a forward proof reconstituted from the tactic justification. The drawback is that the user is only made aware of an unsuccessful case-split at the very end of the proof.

To overcome this drawback, we have implemented two functions `case` and `end`. The `case` function is used to introduce a new case term ϕ. It then generates two subgoals, the first with ϕ as hypothesis, and the second with $\neg\phi$ as hypothesis. The `case` step, therefore, has performed a case-split on $\phi \vee \neg\phi$. The user must first prove the goal on the hypothesis of ϕ. Once the goal is solved, the one remaining goal will have $\neg\phi$ as its hypothesis.

The user now proceeds by introducing the *next* case, using the `case` function again with a new term, say ψ. Two subgoals are again generated, one with ψ and the other with $\neg\psi$ 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, these negations will entail a contradiction[2]. 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 theorem $\phi \implies P \vee Q \vee R$ and suppose that each of P, Q and R can solve a goal G using the implicit automation built into Mizar Light. Then we can write the proof:

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

[2]We assume we are only interested in *classical* proofs. Otherwise, this does not necessarily follow.

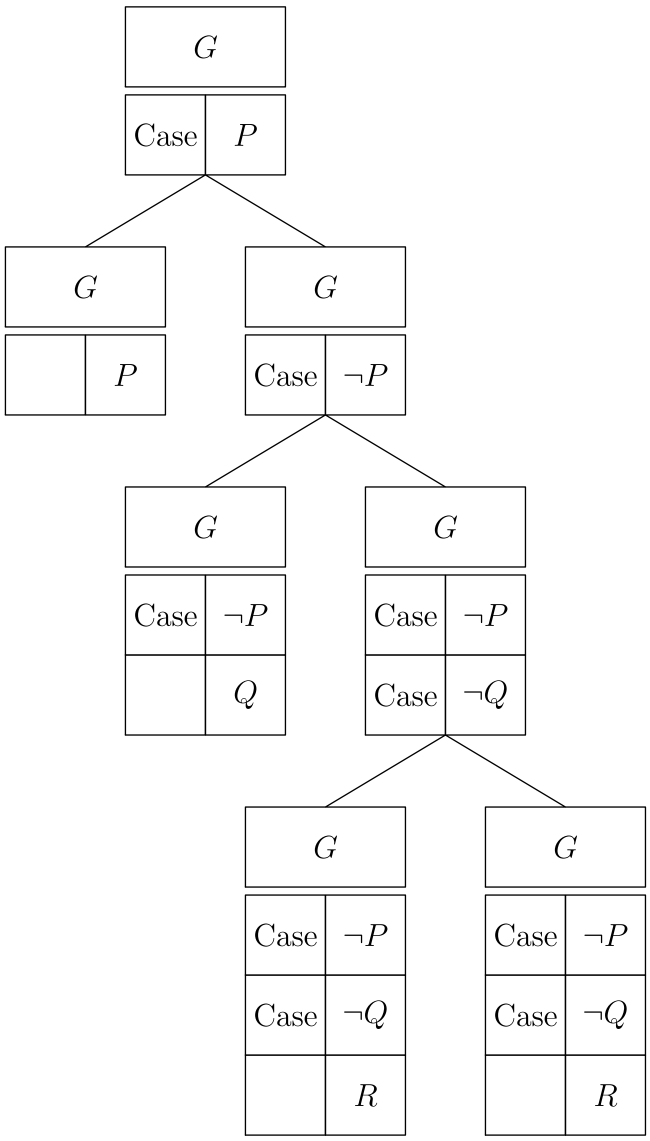


Figure 1.3: Case-splitting Proof Tree

The resulting tree of goal-stacks is depicted Figure 1.3.

The final `end` step is justified since

$$P \lor Q \lor R, \neg P, \neg Q, \neg R \vdash \bot$$

This approach does not have the drawback of Harrison's solution. Whenever the case splitting is not exhaustive, the `end` step will fail at the point at which it is evaluated, rather than at the very end of the proof when the final justification is assembled.

Returning to our example proof, 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
```

1.3.7 Concluding Remarks

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

For clarity, when we exhibit Mizar Light verifications in the rest of the thesis, we shall elide much of the syntax, such as the list delimiters, brackets and semicolons.

Many of our ideas evolved during the proof development, and there are legacy naming conventions which need to be refactored and brought into line with the present thesis. Because of this, the proofs, as of writing, are not identical to the one's we give in this thesis. They are, at least, alpha-equivalent.

Further modifications to Mizar Light are described in Chapter 3.

Chapter 2

Axiomatics

Hilbert has only the briefest introduction to the *Foundations of Geometry*, before diving in with a declaration of his primitive notions and then laying out his first two groups of axioms. These are his axioms of *incidence* and *order*, which together, serve to define *ordered geometry*. In this chapter, we discuss their formalisation, the verification of a few of their elementary consequences, and then discuss how the incidence axioms in particular feature in Hilbert's development.

2.1 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 Hilbert and Pasch both held this sort of abstraction as fundamental to rigour in geometry [80]. As Hilbert was known to remark, it would serve just as well to give the inhabitants of the three sets the names *mugs*, *tables* and *chairs* [37].

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 the theory would work just as well if we only referred to *mugs*, *tables* and *chairs*, and quite another to carry this out. We will not be evaluating the matter, but we would conjecture that it would be very difficult for a human to actually follow the steps of Hilbert's arguments if it was *literally* rendered in terms of *mugs*, *tables* and *chairs*. If Hilbert thought a reader could follow his proofs so blindly, then why did he feel the need to accompany his proofs with diagrams? Diagrams are completely outside the scope of formalised mathematics.

This brings us to the problem with formalised 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 any reference to human intuitions. If this is too much to expect of a human, then it is quite something to expect of a machine, notwithstanding Veblen and Poincaré's claims that Hilbert's arguments would pass through a device as primitive as a Stanley Jevons logical machine [27, 76].

It is perhaps to be expected that we found, along with Meikle and Fleuriot [48], that there are many weaknesses in Hilbert's presentation from the standpoint of formalised mathematics. Hilbert misses out complex proofs, mistakenly declaring them to be easily obtainable. His decisions about when it is necessary to cite a particular axiom are erratic at best. Furthermore, we suggest that much of the time, Hilbert lets intuition dictate what reasoning steps are permissible. In other words, the *documentation* is driving the development. This cannot happen when we do machine verification.

One reason to mention all of these issues is because Hilbert liberally sprinkles his text with documentation that has no bearing on the formal development. Having introduced points, lines and planes, he 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*." We are happy to ignore these comments, and all others like them, without worrying about jeapordising our aims of producing a *faithful* formalisation of Hilbert's axiomatics. These comments, even when they take the superficial form of definitions, are mere *documentation*, no more a part of the formal theory than Hilbert's diagrams.

In general, we shall only introduce new derived terms into our formal theory when they identify useful abstractions.

2.2 Group I

2.2.1 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 said 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 certain standard inference rules (see Mendelson’s classic text [50]).

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 neglected: we will represent each of Hilbert’s three sets by a *predicate*, and consider relativising Hilbert’s axioms to this predicate. 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 sort, and one primitive incidence relation. We leave the sort abstract.

The formalisation is in Figure 2.1. Here, we have formalised four of Hilbert’s incidence axioms (I, 1, I, 3.2, I, 4.1 and I, 8 from Appendix A) as conditions on the predicate sets `point`, `line` and `plane` and the single relation `lie`. We then prove that any four objects satisfying these conditions are such that the three predicate sets are inhabited.

2.2.2 Axioms and Formalisation

Knowing that our types are inhabited, we return to what we regard as a more natural formulation of Hilbert’s axioms, one which appeared in our earlier work, in Meikle and Fleuriot’s work and the work of Dehlinger at el [12, 48, 65]. We declare three primitive *types* for points, lines and planes, and 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 2.1, since we can drop the relativising

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

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

Figure 2.1: Points, lines and planes exist

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 pathological expressions such as "a plane lies on a point."

We now give Hilbert's incidence axioms as they appear in the tenth edition of *Foundations of Geometry*. The convention throughout, and one which we adopt for the rest of the present work, is 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.

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.*

Even these simple axioms have undergone a substantial amount of revision since the first-edition, with axioms being reordered, combined, 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 *Foundations of Geometry*. In fact, only a few interdependencies were ever considered.

We give our formalisation in Appendix A. The theorems in the appendix are asserted as axioms in HOL Light via the function `new_axiom`. Here, we will just give some supplementary discussion of the axioms as of the tenth edition.

Hilbert's Axioms II, 3 and II, 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, require that two points uniquely determine a line. Analogous axioms for planes are given by I, 4.1 and I, 5, while the *converse*, namely that a line is determined by two points appears as 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 was clearly aware that the former could be derived from the latter, but the proof and theorem are both absent from the text. We present our own proof and its formal verification in §2.2.4.

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 §5.3.2, 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 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 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, such as in Axiom I, because there is a line through the point A and B whether or not A and B are distinct. But, we keep the weaker form as the axiom, and verify the stronger version as a theorem.

$$\vdash \exists a. \mathtt{on\_line}\ A\ a \land \mathtt{on\_line}\ B\ a$$

2.2.3 Related Axiomatisations

Hilbert's axiomatisation appears within a culture of related attempts to rigorise geometry. Oswald Veblen's doctoral work [77] 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 [73]. Like Veblen, Tarski used only one primitive sort for points, but unlike Veblen, he admitted a congruence relation on pairs of points. Nevertheless, his 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.

Despite this, 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 proofs in the theory. In fact, it takes significant effort just to recover Hilbert's axioms from Tarski's [3]. There are existing mechanisations of Tarski's geometry in the Otter automated prover [61] and Coq proof assistant [57].

2.2.4 Elementary Consequences

Hilbert highlights two theorems 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.
>
> 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.

These theorems are largely trivial. In fact, the first and last clause in Theorem 1 are barely rewordings of Axiom I, 2 and Axiom I, 6, [1] and so we did not bother to formalise them. The other clauses are only slightly more involved.

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 more suitable for formalising: two planes with a point in common intersect exactly in some line.

$$\begin{aligned}&\alpha \neq \beta \wedge \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ P\ \beta \\ &\qquad\implies (\exists a. \forall Q. \texttt{on\_plane}\ Q\ \alpha \wedge \texttt{on\_plane}\ Q\ \beta \iff \texttt{on\_line}\ Q\ a). \quad (2.1)\end{aligned}$$

The verification of this theorem is given in Figure 2.2. It is a straightforward piece of Mizar light, using five of the ten axioms. This shall be the first piece of Mizar light we will demonstrate, one which we hope makes a convincing case that readable proof scripts are possible in this simple combinator language.

Theorem 2 admits a straightforward formalisation in two separate formal theorems. The verifications depend on Axioms I, 2, I, 3.1, I, 4.1, I, 5 and I, 6.

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

[1]We are assuming classical logic here. In fact, the first clause of Theorem 1 assumes that point equality is decidable. See Dehlinger et al [12].

$$a \neq b \wedge \texttt{on\_line}\ P\ a \wedge \texttt{on\_line}\ P\ b \\ \implies \exists!\alpha.\forall P.\texttt{on\_line}\ P\ a \vee \texttt{on\_line}\ P\ b \implies \texttt{on\_plane}\ Q\ \alpha. \quad (2.2)$$

We omit the verifications, since Hilbert does not refer to these theorems again, nor do their verified counterparts feature anywhere in our formal development. Instead, we base all our future incidence reasoning on an alternative set of representations and theory which we explain in §2.2.6.1.

The only other theorem we need is one we use when developing our theory of half-planes in Chapter 6. This theorem was mentioned earlier in this section: every plane contains a non-collinear triple, an axiom of the first edition but a statement that now needs to be proven. It is perhaps an oversight of Hilbert's that he neglected to state the theorem explicitly, since the proof is not entirely trivial, and has us considering a configuration of six points and three planes.

Theorem. *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 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 ad 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 2.3.

$$\exists A\ B\ C.\texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \\ \wedge \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \quad (2.3)$$

□

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.

```
assume $\alpha \neq \beta \wedge$ on_plane $P\ \alpha \wedge$ on_plane $P\ \beta$                                   1
so consider $Q$ such that $P \neq Q \wedge$ on_plane $Q\ \alpha \wedge$ on_plane $Q\ \beta$ by (I, 7)       2
so consider $a$ such that on_line $P\ a \wedge$ on_line $Q\ a$ by (I, 1)                          3
take $a$
fix $R$
have on_plane $R\ \alpha \wedge$ on_plane $R\ \beta \implies$ on_line $R\ a$ proof
    assume on_plane $R\ \alpha \wedge$ on_plane $R\ \beta$                                        4
    otherwise assume $\neg$on_line $R\ a$                                                5
    hence $\neg(\exists a.$on_line $P\ a \wedge$ on_line $Q\ a \wedge$ 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 2.2: Intersecting planes intersect in a line

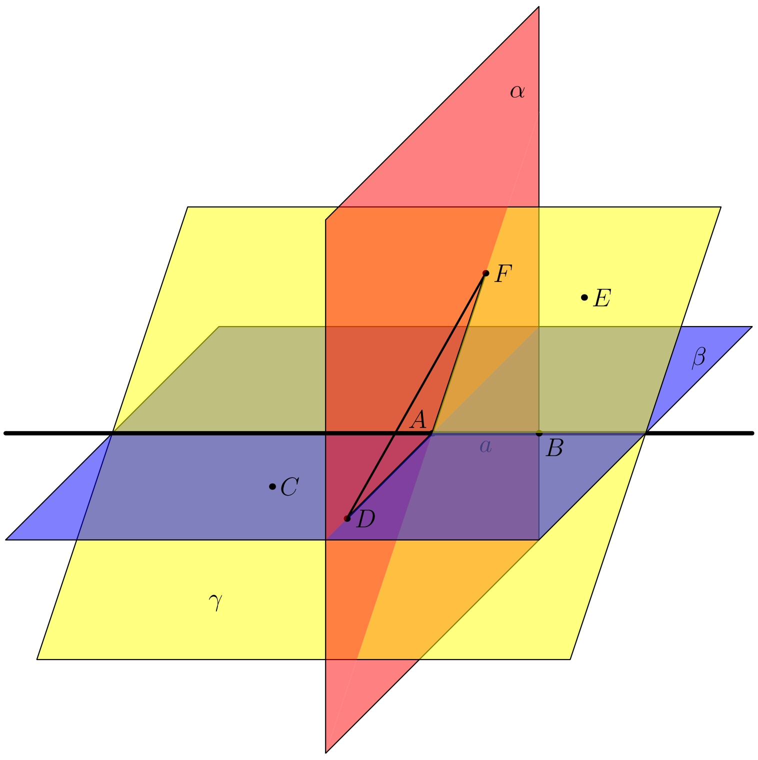


Figure 2.3: Planes contain non-collinear points

2.2.5 Absent Arguments

After splitting conjunctions, we find that there are 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 a claim made by Weyl that the deductions in Hilbert's geometry contain no *gaps* [80]. Indeed, taking his claim at face-value, we would have to conclude with Meikle and Fleuriot that Hilbert's arguments are full of missing assumptions and lemmas. But we do not believe Hilbert ever made this claim. Given that he is happy to elide whole proofs, it seems to us that whatever his standards of rigour, they were never intended to coincide with the extremely pedantic standards of formal verification. Perhaps the only standard we want to 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.

Actually, 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 and I, 3.1.

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 could see no way to do this using Isabelle's module system nor HOL Light's simple type theory. Theory embeddings which depend upon particular *planes* would suggest we would need at least a dependently typed logic such as Coq's [85].

One last reason we might let Hilbert off the hook is by simply claiming that his missing lemmas and missing citations are just *trivial*. If the missing detail happens to end up dominating a formal verification of Hilbert's geometry, then so much for formal verification. It just confirms Poincaré's frank dismissal of the relevance of formal verification to mathematicians that appear in both his review of the *Foundations of Geometry* and his review of Russell and Whitehead's *Principia Mathematicia* [27, 28].

2.2.6 Representation

Whatever position we take on Hilbert's gaps, we still wished to use formal verification as a tool with which to analyse his prose arguments. When we are forced to add the missing detail concerning incidence, our verifications no longer reflect the prose even structurally. The salient parts of the proof text, those parts which correspond to inferences in the prose, are lost to the surrounding lengthy arguments about incidence. This made it difficult to identify dependencies, redundancies, circularities, missing details, or alternative proof strategies.

A first step in the direction of fixing this problem was to find a representation for incidence statements which was significantly more expressive than the basic primitives. Realising that the domain of incidence reasoning is inherently combinatorial, we opted to represent 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.

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

$$\texttt{planar : (point} \rightarrow \texttt{bool)} \rightarrow \texttt{bool}$$
$$\texttt{planar Ps} \iff \exists a.\forall P.P \in Ps \implies \texttt{on\_plane}\ P\ a.$$

Our use of sets here does not depend on the axiom of infinity, since it might still be that only finite sets exists. The use of set theory is really only giving us a way, at the object level, to collapse multiple claims about the incidence of individual points to one claim about a collinear or planar point set, while abstracting away the particular lines and planes with which the points are incident.

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 express the fact that two

$$\texttt{collinear}\ \{A,B\} \tag{2.4}$$

$$S \subseteq T \wedge \texttt{collinear}\ T \implies \texttt{collinear}\ S \tag{2.5}$$

$$A \neq B \wedge A,B \in S,T \implies \texttt{collinear}\ S \wedge \texttt{collinear}\ T \implies \texttt{collinear}\ (S \cup T) \tag{2.6}$$

$$\texttt{planar}\ \{A,B,C\} \tag{2.7}$$

$$S \subseteq T \wedge \texttt{planar}\ T \implies \texttt{planar}\ S \tag{2.8}$$

$$\neg\texttt{collinear}\ (S \cap T) \wedge \texttt{planar}\ S \wedge \texttt{planar}\ T \implies \texttt{planar}\ (S \cup T) \tag{2.9}$$

$$\texttt{collinear}\, S \implies \texttt{planar}\, S \tag{2.10}$$

$$A \neq B \wedge A,B \in S,T \wedge \texttt{collinear}\ S \wedge \texttt{planar}\ T \implies \texttt{planar}\ (S \cup T) \tag{2.11}$$

$$P \in S \wedge P \in T \wedge \texttt{collinear}\ S \wedge \texttt{collinear}\ T \implies \texttt{planar}\ (S \cup T) \tag{2.12}$$

Figure 2.4: Incidence Rules in Point Sets

other points C and D also lie on a by writing

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

Furthermore, we can say that another point E does *not* lie on the line a by writing $\neg\texttt{collinear}\ \{A,B,E\}$. Notice that we only need to use three points to make this claim, and that indeed, adding more points only weakens the statement (collinear sets which contain only three-point collinear sets are themselves collinear). This means that claims of non-incidence are claims about the existence of *triangles*.

The real advantage to be gained by talking in terms of collinear and planar sets is that we were able to move all the logic of incidence into composition rules for sets. The systematic use of these rules made up the bulk of the missing incidence reasoning in our earlier work [65].

2.2.6.1 Incidence Reasoning with Point Sets

In Figure 2.4, we reproduce a set of rules for reasoning with collinear and planar sets which were used in our earlier work in Isabelle [65], and which we have verified again in HOL Light. 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 rules, we can make these axioms correspond to the incidence axioms which do not introduce points, and thus justify them as an alternative presentation of the incidence rules. We regard this as a somewhat important observation, since we do not want just a bunch of *ad-hoc* rules which happen to work most of the time. We almost want an alternative way to think about incidence reasoning, one which has nice computational properties. We can then offer the idea that Hilbert omitted incidence arguments *because* they are just computational detail.

This is perhaps overly fussy, but we want our verifications to reflect a more *charitable* reading of Hilbert than appears in Meikle and Fleuriot [48]. We encourage readers who find these questions rather bland to skip ahead.

Given a 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, Rule 2.4 must assert that the points A and B are incident with a line AB, corresponding to Axiom I, 1 [2]. Rule 2.6 is less easy to understand, but effectively corresponds to Axiom I, 2. It asserts that the expression "the line AB" is well-defined.

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 Rule 2.6 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, Rule 2.7 tells us that A, B and C are incident with the plane ABC while Rule 2.9 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.

Rule 2.11 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 2.10, S is also planar, and according to 2.11, so is $S \cup T$. 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.

Our final rule, Rule 2.12, tells us that, once we understand lines to be maximal collinear

[2]A nice way to read this rule is to say that when two lines meet at *more* in more than two places, they must form a straight line rather than a hinge. Thanks to Robert Henderson for spotting this.

sets, then intersecting lines lie in a plane. But if these lines are distinct, their union must be non-collinear, since otherwise they would not be maximal sets. Thus, the two lines lie in a unique plane, which is exactly the claim made in Hilbert's Theorem 2.

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 Rule 2.6. 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 Rule 2.9. Moreover, we know that if the intersection contains two points, then, according to rules 2.6 and 2.11, the two points yield a maximal collinear set contained in the maximal planar set. We cannot 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.

In conclusion, we have shown how all incidence axioms which do not introduce points can be expressed and strengthed as composition rules on collinear and planar sets. The primitives *line* and *plane* are no longer used directly, since all the axioms governing them can be subsumed by these composition rules.

2.2.6.2 Evaluation

Meikle and Fleuriot's work in 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 our alternative rules, the verifications are a good deal less complex. For instance, the verification of Hilbert's third theorem, which relies heavily on incidence reasoning, needed twenty-seven special case lemmas and a forty step proof in Meikle's verification, while our proof using the above rules has twenty-two steps and no additional lemmas. We will give some examples of these proofs in Chapter 4.

This is an improvement, but it still leaves our proofs bogged down in trivial combinatorial details that make it difficult to compare to the prose. Besides, the proofs are still very difficult to obtain, since the sets involved in the rules in Figure 2.4 almost always need to be manually instantiated so that the theorem prover's generic automation can make progress. This is not just tedious, but error-prone. When our proofs were correct, they were often suboptimal. And in one case, our difficulty in proving certain incidence facts about a geometric configuration led us to believe, mistakenly, that Hilbert had made an error in one of his arguments (see §4.3.1.1).

Luckily, when we reflected on our manually crafted proofs, we realised that the rules of Figure 2.4 were always applied systematically. In the next chapter, we shall describe how we made these rules the basis for an automated tool which can completely hide the messy incidence reasoning. We can then justify Hilbert's omission of incidence arguments by claiming that they are mostly computational and do not require any real insight. In a geometric proof, the important rules 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 rules explicit.

We finish by remarking that the rules of Figure 2.4 were verified in a procedural rather than declarative style in HOL Light. We only intend these verifications to justify rules that will be become the implementation details of an incidence algorithm, rather than rules 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.

2.3 Group II

Hilbert's second group of axioms give us a bare *ordered geometry*, which defines the scope for the present work. With such few geometrical notions, things might seem quite restrictive, but there is a good history of interest in systems where we have these "axioms without measurement" [59].

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, and to this day, Hilbert's one planar axiom in this group — and its variants — 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 [31], while E.H. Moore and his student Veblen showed how, via Pasch's axiom, Hilbert's main linear axiom was derivable. Veblen showed great interest in a bare ordered geometry, proving forty theorems compared to Hilbert's seven. He gave an early proof attempt of the polygonal Jordan Curve Theorem (see Chapter 7), and later set out to recover the full metrical Euclidean geometry using only order axioms, the parallel axiom and a continuity axiom.

2.3.1 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 in the group

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.*

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:

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

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 `point` → `point pair` → `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 consquence would be that other axioms and theorems would sometimes have to make additional non-degeneracy assumptions.

Our formalised rendering is just slightly weaker than Hilbert's. As Veblen spotted [77], we will not need to assume that all points are distinct.

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

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.*

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.

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 in Appendix B with that for the other axioms.

2.3.2 Pasch and Incidence Reasoning

In fact, in our earlier work, most of our effort was expended trying to verify the preconditions of Pasch's Axiom (II, 4). In order to leverage our representation in point sets, we decided to rewrite 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 and the segment AB, and the point E will be any other point on the line a. See Figure 2.5.

Our preconditions can now be expressed in terms of non-collinear and planar sets. We do this by deriving the following alternative formulation of Axiom II, 4:[3]

$$\neg\texttt{collinear}\ \{A,B,C\} \wedge \neg\texttt{collinear}\ \{A,D,E\} \wedge \neg\texttt{collinear}\ \{C,D,E\} \tag{2.13}$$

$$\wedge\, \texttt{planar}\ \{A,B,C,D,E\} \wedge \texttt{between}\ A\ D\ B \tag{2.14}$$

$$\implies \exists F.\texttt{collinear}\ \{D,E,F\} \wedge (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C). \tag{2.15}$$

[3]Thanks to Laura Meikle for spotting that we only need to assume three triangles here.

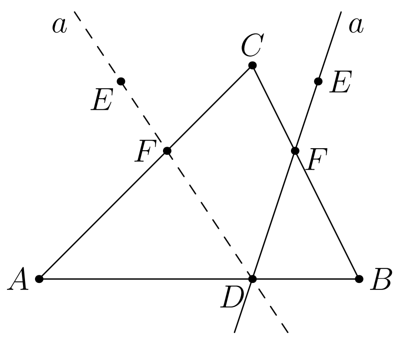


Figure 2.5: Axiom II, 4

It is now perhaps clearer how the rules from §2.2.6.1 are needed when reasoning about incidence in Hilbert's proofs. Most of his theorems 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 Theorem 2.13, 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 arguments from our earlier work where laden down with proofs to find triangles by repeatedly applying rules from §2.2.6.1. We will show some of the complexity involved here in Chapter 4, but will see how, luckily, it can all be easily automated.

2.4 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 inhabited. Otherwise, Group I is conspicuous only by its absence from the remaining formal theory, especially in Group II where incidence reasoning is needed heavily in order to apply Pasch's axiom. Whatever way we justify the absence of missing detail, it must be restored in the formal verification, and when we do that, we find our proofs are washed out with fussy incidence arguments. The skeleton of Hilbert's proofs are lost to the details.

By introducing better representations, we can alleviate this somewhat, but it only takes us so far. If we are to make Hilbert's arguments clear in our verifications, so that we can better use verification to judge and evaluate his prose proofs, we shall need to remove the incidence arguments almost entirely. This we leave to the next chapter, where we consider automation.

Chapter 3

Automation

In the last chapter, we considered a representation for incidence claims involving lines and planes which helped us write shorter verifications. The use of these rules also appeared ripe for automation.

The HOL Light theorem prover expects its users to work “close to the metal”, often writing ML directly. Writing new tools can be comfortable in this environment, and it is easy to prototype and experimentally integrate new tools into the rest of the system and existing proof languages. In this chapter, we shall see how this integration can take the form of a concurrent and collaborative automated tool for incidence reasoning, which can be made available as a tactic and in Mizar light proof steps.

Much of this chapter and the subsequent chapter is an expansion and improvement on earlier work [66, 67].

3.1 Related Work

As far as fully automated theorem proving goes, the oldest successes are probably in geometry. The signed-area method [69] of Chou et al, proved to be particularly capable at finding large numbers of non-trivial geometric theorems. But in this section, we restrict our attention to automation that can apply to the very basic incidence theory we consider here. The signed-area method, which typically assumes an ordered field, is already outside of our scope.

3.1.1 Wu's Method

When considering general automation in Hilbert's *Foundations of Geometry*, there is probably no work more relevant than Wu's method [74]. Wu credited the *Foundations of Geometry* for the metatheoretical insights that led to his mechanisation procedure for the whole of geometry[1].

One of Wu's own 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 theorems for some geometric figure, we have in mind a particular "genericity" of that figure which is hard to capture formally. Moreover, the axioms and theorems 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 in the last group that some of Hilbert's axioms, such as Axiom I, 1 hold in degenerate cases (though this does not imply that the axiom should be made *stronger*). We shall show in Chapter 6 how the particularly troublesome Axiom II, 4 (troubling at least as far as incidence reasoning is concerned) can be strengthened by allowing degenerate cases.

But "degenerate" is not well-defined, and it is not always clear how the conditions of a theorem 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 proof, so it would seem that Wu is correct: we cannot be truly rigorous unless we can systematically deal with degenerate conditions. This provides an alternative way to diagnose the gaps in Hilbert's proofs spotted by Meikle and Fleuriot [48]: they were gaps concerning degeneracy conditions about point incidence.

Wu's highly celebrated method automatically inserts non-degeneracy conditions to the theorems it proves, and if possible, it automatically deletes redundant conditions to make a specific theorem more generic. The method has been used to automatically prove an enormous number of non-trivial theorems in unordered geometry [68]. To cover ordered geometry, Wu appealed to the embedding of Euclidean geometry in real-closed fields, for which Tarksi has a well-known decision procedure. The method has much less success here, owing to the gross intractability of the decision procedure [46].

Unfortunately, Wu's automation is not particularly appropriate for our own work. If

[1]Wu was possibly giving Hilbert too much credit

we were to exploit it, 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 theorems which we are trying to mechanise. Indeed, Wu's method is based at least on results which rest on Desargues' theorem — Theorem 53 in the *Foundations of Geometry*. Furthermore, our focus in the present work is *ordered geometry*, a domain in which Wu's method is ill-suited.

3.1.2 Ranks

For the specific problem of incidence reasoning, Magaud et al's work [43] is closest in spirit to our own. It too, is based on point-sets, but the authors have beautifully abstracted the core idea.

Several of our rules for lines and planes in Figure 2.4 in the last chapter appear analogous. Magaud et al have captured the analogy formally using *ranks*. The rough idea is that 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 our rule for taking the union of collinear and planar sets.

Magaud's approach allows the author's 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.

3.2 Inference Rules

The rules from Figure 2.4 tell us how to reason with finite collinear and planar sets, and so can form the basis of a combinatorial algorithm. Our rules trade in four kinds of theorem: theorems about which points are distinct, which points are collinear, which triples are non-collinear, and which points are planar. These theorems will be the domain of our algorithm, whose chief procedures will be based on rules to interderive them.

Our rules already show how to introduce collinear and planar sets. Additionally, we need ways to derive the inequalities and triangles on which the rules depend. 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 simple congruence reasoning: 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 on patterns of reasoning that showed up in our Isabelle formalisation. 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 rules. Again, we followed a pattern of argument that showed up in the Isabelle formalisation. 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 and methods for introducing new theorems.

`ncolneq` Infer inequalities from non-collinear triples:

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

(by rule 2.4);

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

$$\begin{aligned}&\texttt{collinear}\ S \wedge \neg\texttt{collinear}\ \{A,B,C\}\\ &\quad \wedge A,B \in S \implies \forall S. S \in As \implies C \neq S\end{aligned}$$

(by rule 2.5). For example,

$$\begin{aligned}&\texttt{collinear}\{A,B,C,D,E\} \wedge \neg\texttt{collinear}\{A,B,P\}\\ &\qquad\qquad \implies C \neq P \wedge D \neq P \wedge E \neq P.\end{aligned}$$

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

$$\begin{aligned}&\texttt{collinear}\ S \wedge \texttt{collinear}\ T \wedge \neg\texttt{collinear}\ U \wedge U \subseteq S \cup T\\ &\quad \wedge A,B \in S,T \implies A = B\end{aligned}$$

(by rules 2.5 and 2.6). For example,

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

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

$$\texttt{collinear}\ S \land \neg\texttt{collinear}\ \{A,B,C\}$$
$$\land X,Y,A,B \in As \land A \neq B \implies \neg\texttt{collinear}\ \{C,X,Y\}.$$

(by rules 2.5 and 2.6). For example,

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

`colcol` Use rule 2.6 to show that the union of collinear sets which intersect at more than one point is collinear.

`planeplane` Use rule 2.9 to show that the union of planar sets intersecting at a non-collinear triple is planar.

`colplane` Use rule 2.10 to show that a collinear set is planar.

`colplaneplane` Use rule 2.11 to show that the union of a collinear and planar set intersecting in at least two points is planar.

`colcolplane` Use rule 2.12 to show that the union of intersecting collinear sets is planar.

3.3 Näive Implementation

Our first prototype used basic forward-chaining, and such methods have been used with some success to investigate Hilbert's geometry [71]. Forward chaining seemed particularly suitable for our use case. We have already noted that some of our proofs were suboptimal, that we believed Hilbert might have made errors in assuming certain non-degeneracy conditions, and so we wanted a tool which could *explore* 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 proof. Forward-chaining seemed quite apt, since its focus on continually growing a base of facts corresponds well with how a declarative proof is developed by continually growing a proof context.

Our first algorithm worked on *generations* made up of five lists, one for each of the different kinds of fact: point equalities, point inequalities, collinear sets, non-collinear sets, and finally planar sets. The procedures of §3.2 were then applied across all relevant combinations of theorems from the five lists, to produce a new generation of lists. In this way, the algorithm produces a sequence of generations, each a superset of its predecessor.

3.3.1 Concurrency

Typically, the automation available in interactive theorem provers 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 Theorem prover has capitalised on this with Sledgehammer [51]. 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, since it is almost trivial to turn our algorithm into something of a "collaborative" architecture. We are focusing on forward-chaining and declarative proof, and as we remarked, the way that a user builds a declarative proof is by growing a proof context towards a goal, while a forward-chaining algorithm interatively grows a set of facts. These processes are analogous, so why not splice the two? A user's proof script is then a user guided, manually crafted search through the proof space, but now one which has access to the derivations of a forward chaining algorithm. At the same time, the forward chaining algorithm works independently, but can freely incorporate the user's manually obtained deductions into its own base of facts. The two systems, the automation and the human user, can thus be seen as collaborating, sharing data as they both strive towards the goal.

We started with a very simple implementation of this setup. We spawn a thread that is initialised with an empty generation of facts. New generations are produced iteratively through forward-chaining. At the same time, the thread monitors the proof context, so that each time the user adds a hypothesis by performing a declarative proof step, the hypothesis can be automatically inserted into the current generation and used for further chaining.

The thread then echoes each generation of facts concurrently to a separate terminal, while the user works on the proof script. We then introduce a variable `the_facts` which is updated so that it always dereferences to the current generation. This variable can be used in a proof script to justify the current step with the automatically inferred facts.

3.3.2 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. These make its purposes so modest that it is not clear why such a tool would ever be needed. Perhaps it would be useful to set the scene.

We would typically develop our geometric proofs on paper, but as we noted above, a typical geometric diagram probably includes many non-degeneracy assumptions. 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, which we would again trace out on paper.

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 theorems 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 results from the search process, including results we had not expected, and that we generally did not run the searches with a particular goal theorem 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 theorems and concepts without the need for assistance (see the work of Colton et al [8], as well as McCasland and Bundy [45]).

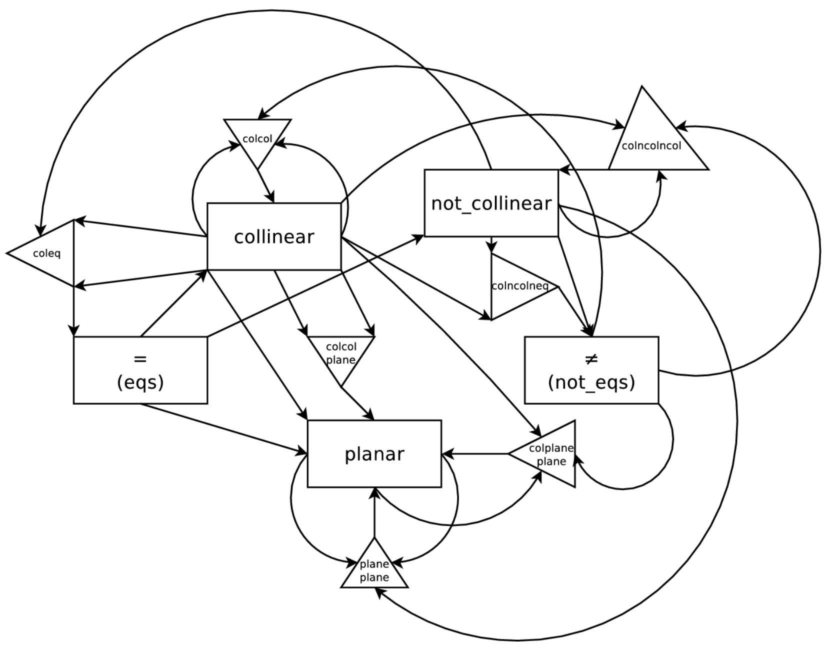


Figure 3.1: Data-flow for incidence reasoning: boxes represent our collections of various kinds of theorem, while triangles represent inference rules

3.4 An Implementation in Combinators

3.4.1 Data-Flow

Our prototype implementation was an adequate proof of concept, and enabled us to write some very short and clear proofs. We have some examples of these in the next chapter. However, our prototype currently has no theoretical underpinning. The finer details of the implementation are complex, ad-hoc, unlikely to scale, and cannot be easily modified to work on new problems.

Much of the complexity of the ML algorithm comes from the fact that each generation of theorems is partitioned into five types: point equalities, point inequalities, collinear sets, non-collinear sets and planar sets. By partitioning in this way, we greatly reduce the number of combinations of theorems we need to try with each inference rule, but there is still much wasted effort. Every time a new generation is generated, all relevant combinations are applied, *including* the combinations which were applied for the *last* generation. Our simple lists do not help us filter out the repetitious work.

Rather than seeing the problem as simply one of repeating inferences across a succession of generations, we decided to model the data-flow directly. The flow of data for our problem, according to the rules of §3.2, is given in Figure 3.1.

The particular automation we need for Hilbert lies in a very narrow domain, and we would rather see it as a “one-shot” tool, one which is unlikely to be needed in other projects. This makes it most similar in purpose to a hand-crafted conversion or tactic [53]. Conversions and tactics, a staple of the LCF approach to theorem proving, are normally written and understood as combinators [72], which are themselves a staple of strongly-typed functional programming.

Our aim in this section will be to show how a data-flow such as that of Figure 3.1 can be directly expressed in a suitable combinator language, with a distinguished set of primitives which can be combined by transformations. While the incidence automation itself will be a one-shot tool written in this language, we hope the data-flow language will be more useful in other contexts.

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 and can easily inject computations into it using lambda abstraction.

3.4.2 Related Work

The idea of an algebraic data-flow language was considered early on by Chen [5], 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 [38]. The algebra we shall consider here is related and was originally conceived by Spivey [70]. It has more rigorously developed underpinnings, and compared to other search algebras such as those of the logic programming monad [38], it places stronger constraints on the order in which the values are generated. We shall be generalising Spivey’s ideas somewhat, and also hope to give a more “operational” motivation for the definitions.

3.4.3 Streams

Our overarching purpose is to output a stream of theorems, perhaps to a terminal, to a database of facts to be used during a proof, or perhaps to another consumer for further processing. If we think of this output *as* the implementation, then we are dealing with procedures which lazily generate successive elements of a list. These lazy lists, or

$$\begin{aligned}
&\texttt{fmap}\ (\lambda x.\ x)\ m = m \\
&\texttt{fmap}\ f \circ \texttt{fmap}\ g = \texttt{fmap}\ (f \circ g) \\
&\texttt{fmap}\ f \circ \texttt{return} = \texttt{return} \circ f \\
&\texttt{fmap}\ f \circ \texttt{join} = \texttt{join} \circ \texttt{fmap}\ (\texttt{fmap}\ f) \\
&(\texttt{join} \circ \texttt{return})\ m = m \\
&(\texttt{join} \circ \texttt{fmap}\ \texttt{return})\ m = m \\
&\texttt{join} \circ \texttt{join} = \texttt{join} \circ \texttt{fmap}\ \texttt{join} \qquad (3.1)
\end{aligned}$$

Figure 3.2: The Monad Laws

streams, 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 echos a list of precomputed theorems; it might generate theorems using input data; it might generate them from some other automated tool. We shall focus instead on transformations for streams, and in how we might lift typical symbolic manipulation used in theorem proving to the level of streams.

One reason why lists and streams are a good choice is that they are *the* ubiquitous data-structure of ML and its dialects, and have rich interfaces to manipulate them. A second reason why lists 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 [79]. We can interpret this monad as decorating computations with non-deterministic 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 together with 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 3.2.

The list monad can be used for search, but takes list concatenation `concat` $: [[\alpha]] \to [\alpha]$

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

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

Figure 3.3: Shifting

for its join operation. This makes it unsuitable for *fair* and *unbounded search*. If the list 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 an undesirable property. We want to be able to combine unbounded searches over infinite domains without losing any data.

3.4.4 The Stream Monad

There is an alternative definition of the monad for streams (given in Spivey [70]) which handles unbounded search. Here, the join function takes a possibly infinite stream of possibly infinite streams, and produces an exhaustive enumeration of *all* elements. We show how to achieve this in Figure 3.3 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 searches, then the outer stream 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 results into a single discoverer. The highlighted column in Figure 3.3

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

is this combined result: a set of values generated *simultaneously* and thus having no specified order (this is required to satisfy Law 3.1 in Figure 3.2).

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 §3.3. 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 discoverers of generations assumes certain algebraic laws on them: 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 discoverers must then have type $[G\,[G\,\alpha]] \to [G\,\alpha]$, sending a discoverer of generations of discoverers into a discoverer of generations of their data.

We now denote the k^{th} element of the argument to `join` by $gs_k = \{d_{k,0}, d_{k,1}, \ldots, d_{k,n}\} : G\,[G\,\alpha]$. Each $d_{k,i}$ is, in turn, a discoverer stream $[g^k_{i,0}, g^k_{i,1}, g^k_{i,2}, \ldots] : [G\,\alpha]$. We invert the structure of gs_k using `transpose` $: M[\alpha] \to [M\,\alpha]$, which we can define for arbitrary monads M. This generality allows us to abstract away from Spivey's bags and consider more exotic inner data-structures. We choose the name "`transpose`", since its definition generalises matrix transposition on square arrays (type $[[\alpha]] \to [[\alpha]]$):

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

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 3.4), and thus, the shift function of Figure 3.3 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 discoverers now constitutes a monad (see Spivey [70] 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 discoverers, and combine one discoverer $d : [G\,\alpha]$ with another discoverer $d' : \alpha \to [G\,\alpha]$ which depends, via arbitrary computations, on each individual element of d.

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

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

Figure 3.4: Transpose

of empty generations.

3.5 Case-analysis

Our algebra allows us to partition our domain into discoverers according to our own insight into a problem. For incidence reasoning, we divided the domain into five kinds of incidence theorem. We can then compose our discoverers in a way that reflects the typical reasoning patterns found in the domain.

However, when it comes to theorem-proving, the sets of theorems are further partitioned by branches on disjunctive theorems. In proof-search, when we encounter a disjunction, we will want to branch the search and associate discovered theorems in each branch with its own disjunctive hypothesis.

Ideally, we want to leave such case-splitting as a book-keeping issue in our algebra, and so integrate it into the composition algorithm. Streams must then record a context for all of their theorems, and this context must be respected as discoverers are combined.

Luckily, there is some flexibility in Spivey's search monad. We can choose a data-structure other than bags for the generations. To solve the problem of case-analysis, we have chosen to implement the generations as *trees*.

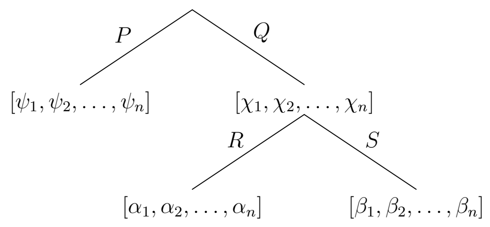

P
Q
$[\psi_1, \psi_2, \ldots, \psi_n]$
$[\chi_1, \chi_2, \ldots, \chi_n]$
R
S
$[\alpha_1, \alpha_2, \ldots, \alpha_n]$
$[\beta_1, \beta_2, \ldots, \beta_n]$

Figure 3.5: A Simple Tree Branching on Case-splits

3.5.1 Trees

Each tree represents a generation of discovered theorems and replaces each of Spivey's bags. Each node in a tree is a bag of theorems discovered in that generation under a disjunctive hypothesis. Branches correspond to case-splits, with each branch tagged 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. Our goal is to discover theorems which hold when the case-splits are eliminated.

Thus, the tree in Figure 3.5 might represent a formula made by case-analysing $P \vee (Q \wedge (R \vee S))$:

$$\phi_1 \wedge \phi_2 \wedge \cdots \wedge \phi_n \wedge (P \implies \psi_1 \wedge \psi_2 \wedge \cdots \wedge \psi_n) \wedge \begin{pmatrix} Q \implies \chi_1 \wedge \chi_2 \wedge \cdots \chi_n \wedge (R \implies \alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n) \\ \wedge (S \implies \beta_1 \wedge \beta_2 \wedge \cdots \wedge \beta_n) \end{pmatrix}.$$

3.5.1.1 Combining Trees

The principal operation on trees is a sum function which is analogous to the append function for lists, combining all values 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. See Figure 3.6.

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 a theorem is deleted from the tree, it must be because it is trivially implied by other theorems in the tree. Our weaker and more wooly constraint is that the topologies should represent a reasonably efficient partitioning of

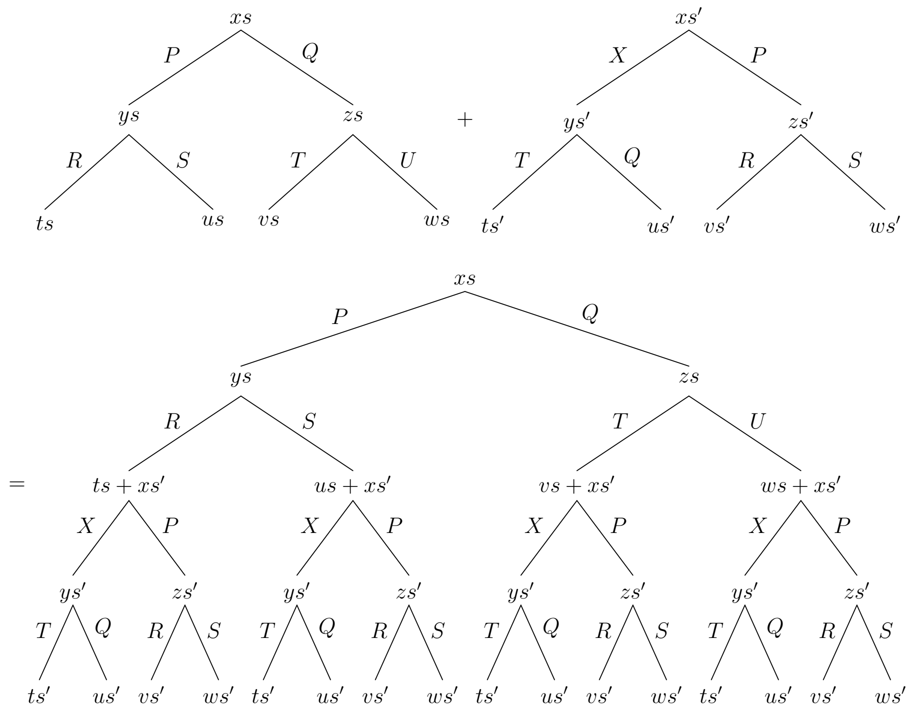


Figure 3.6: Combining Trees by Nesting

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 $P \implies \cdots \implies P \implies \phi$. 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 3.7, where we have coloured duplicate hypotheses along root paths. In the result, we fold the marked subtrees into their parents, and discard the siblings. Notice that, at this stage, no theorems have been lost.

Our next simplification allows us to delete a subtree t if its branch case is already considered by an uncle 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 3.8. The P branch on 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 3.8, 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.

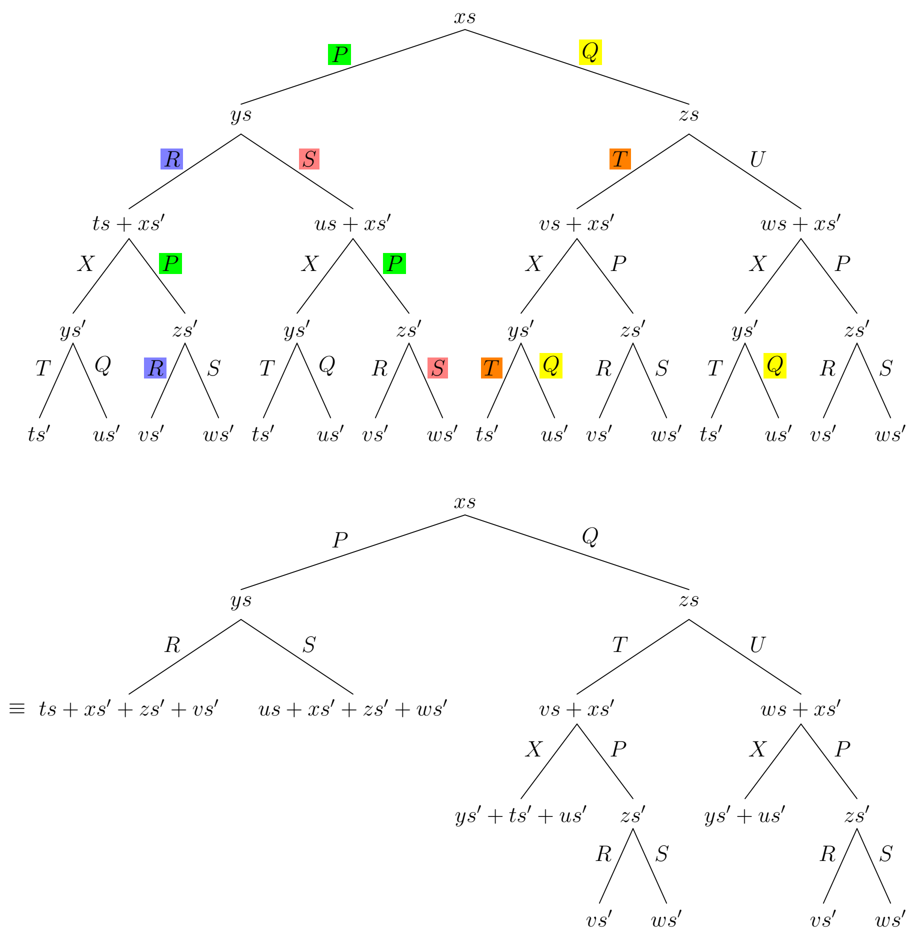


Figure 3.7: Weakening

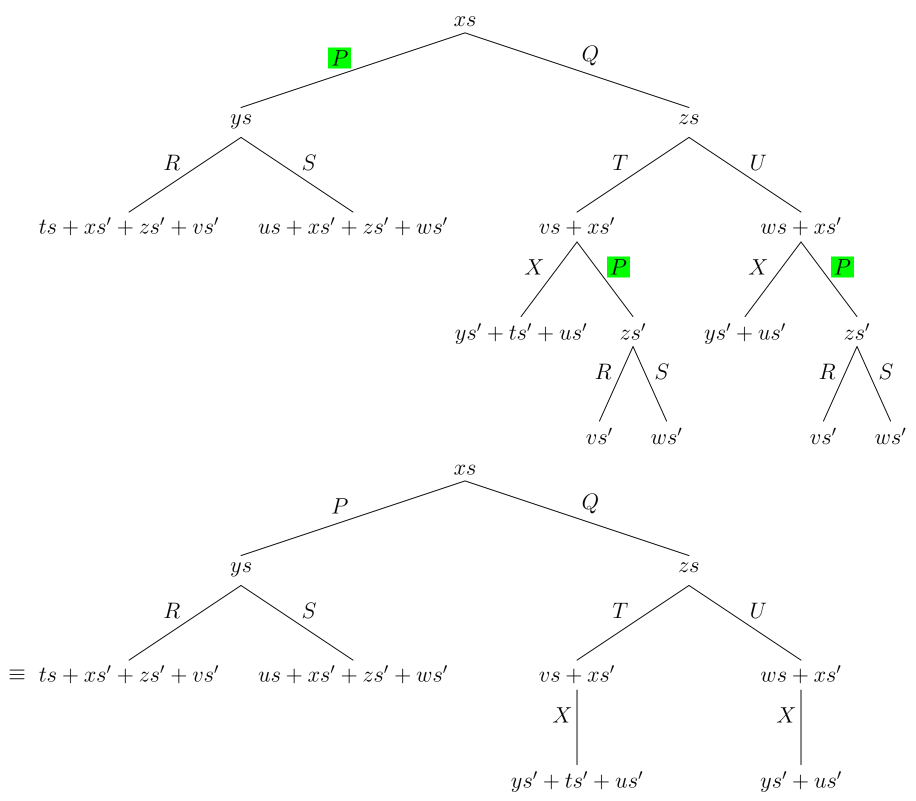


Figure 3.8: Removing Redundant Case-splits

Our final tree is shown in Figure 3.9. We describe our simplification in several steps, though they can be combined into a single pass of the left hand tree. With a single pass, we do have one limitation: we cannot promote all data. If we could, the replicated us' bag in the bottom right branches of the tree could be promoted into the Q branch. We will not elaborate on this implementation issue.

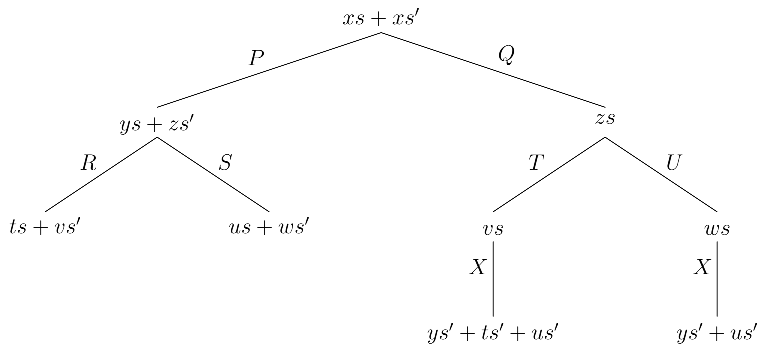


Figure 3.9: Promoting Common Data

3.5.1.2 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 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 conjunction, giving a new tree t' with the *new* type `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 theorems which respect disjunctive hypotheses from their place in t and from their respective inner-trees.

3.6 Additional Primitives and Derived Discoverers

Once we replace Spivey's bags with our trees, we have a stream data-type to represent a theorem discoverer searching a space of theorems which have been partitioned according to case-splits. In fact, discoverers generalise over whatever type of values they are discovering, and so we give them the polymorphic type `discoverer` α.

As we described in §3.4.4, these discoverers form a monoid. Operationally, the sum function runs two discoverers in parallel, respecting their mutual case-splits, while our identity discoverer generates nothing but empty generations.

3.6.1 Case-splitting

Case-splits are introduced by `disjuncts`, which is a discoverer parameterised on arbitrary theorems. Here, `disjuncts` $(\vdash P_0 \vee P_1 \vee \cdots \vee P_n)$ outputs 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 theorem $P_i \vdash P_i$. This process can be undone by flattening trees using `flatten`, which discharges all tree labels and adds them as explicit antecedents on the theorems of the respective subtrees.

3.6.2 Delaying

A generated value 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 streams. If values of type α are generated in a stream from values of type β in another stream, and *vice-versa*, then one of the two streams must be delayed to avoid deadlock.

3.6.3 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 as:

$$\begin{aligned} xs >\!\!>\!= f &= \texttt{join}\,(\texttt{fmap}\, f\, xs) \\ \texttt{filter}\, p\, xs &= xs >\!\!>\!= (\lambda x.\, \texttt{if}\ \ p\, x\ \ \texttt{then}\ \ \texttt{return}\, x\ \ \texttt{else}\ \ 0) \end{aligned}$$

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

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 don't "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 a "subsumption" function in the form of 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).

3.6.3.1 More Sophisticated Filtering

There are two pieces of missing functionality. Suppose we have theorems $\vdash \phi$ and $\vdash \psi$ where $\vdash \phi \leq \vdash \psi$ and a discoverer $xs = [\{\vdash \phi\}, \{\vdash \psi\}]$. That is, xs is a discoverer which produces two generations of theorems. In the first generation is the single theorem $\vdash \phi$ and in the second is the single theorem $\vdash \psi$. Now suppose we have a monotonic function f which happens to have the mappings

$$\vdash \phi \mapsto \vdash [\{\},\{\},\{\},\{\},\{\phi^*\}]$$
$$\vdash \psi \mapsto \vdash [\{\psi^*\},\{\}]$$

Here, ψ is sent immediately to a new value ψ^*, while the image ϕ^* of ϕ takes some time to generate, represented here by a succession of four empty generations. Thus,

$$xs >>= f = [\{\psi^*\},\{\},\{\},\{\},\{\phi^*\}].$$

What happens when we want to take the maxima of this result? 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 discoveres 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 x in the first n generations of xs, there should exist some y in the first n generations of ys such that $x \leq y$. Thus, ys generates facts which are at least as strong as those of xs, and does so at least as early in terms of the number of generations.

Thus, if by monotonicity we require that $[\{\},\{\},\{\},\{\},\{\phi^*\}] \leq [\{\psi^*\},\{\}]$, then we further require that $\phi^* \leq \psi^*$. Therefore:

$$\texttt{maxima}\, xs >>= f = \texttt{maxima}\ ([\{\psi^*\},\{\},\{\},\{\},\{\phi^*\}]) = [\{\psi^*\},\{\},\{\},\{\},\{\}].$$

And here we have a problem: the delayed and potentially slow computation of $\vdash \phi^*$ was wasted. Once $\vdash \psi$ was discovered, further expansion of the discoverer which depends on $\vdash \phi$ 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 2.6 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, ongoing discovery based on any unnormalised term should be abandoned.

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

This additional functionality has not yet been implemented, and might well require significantly modifying the underlying data-structures for our streams. We leave such possibilities to future work.

3.6.4 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 values of a stream, starting with a base-value, and folding each generation down to a single intermediate value. 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 facts discovered so far in a single collection. If the collection is lists, we just use $\texttt{accum}\,(\lambda xs\,x.\,x :: xs)\,[\,]$.

3.6.5 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. In our theorem-prover, the Modus Ponens inference rule can throw an exception, 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.

$$\begin{aligned}\texttt{fmap}'\,f\,xs &= xs \mathbin{>>=} (\lambda x.\,\texttt{try return}\,(f\,x)\ \texttt{with}\ \_ \ \rightarrow 0)\,xs\\ \texttt{fmap2}'\,f\,xs\,ys &= \texttt{fmap}\,f\,xs \mathbin{>>=} (\lambda f.\,\texttt{fmap}'\,f\,ys)\\ \texttt{fmap3}'\,f\,xs\,ys\,zs &= \texttt{fmap}\,f\,xs \mathbin{>>=} (\lambda f.\,\texttt{fmap2}'\,f\,ys\,zs)\end{aligned}$$

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

$$\begin{aligned}
\texttt{chain1}\ \mathit{imp}\ \mathit{xs} &= \texttt{fmap2}'\ \texttt{MATCH\_MP}\ (\texttt{return}\ \mathit{imp})\ \mathit{xs} \\
\texttt{chain2}\ \mathit{imp}\ \mathit{xs}\ \mathit{ys} &= \texttt{fmap2}'\ \texttt{MATCH\_MP}\ (\texttt{chain1}\ \mathit{imp}\ \mathit{xs})\ \mathit{ys} \\
\texttt{chain3}\ \mathit{imp}\ \mathit{xs}\ \mathit{ys}\ \mathit{zs} &= \texttt{fmap2}'\ \texttt{MATCH\_MP}\ (\texttt{chain2}\ \mathit{imp}\ \mathit{xs}\ \mathit{ys})\ \mathit{zs} \\
\texttt{chain}\ \mathit{imps}\ \mathit{xs} &= \mathit{imps} >>= (\lambda \mathit{imp}.\ \texttt{if is\_imp}\ \ \mathit{imp} \\
&\qquad \texttt{then chain}\ (\texttt{fmap}\ (\texttt{MATCH\_MP}\ \mathit{imp})\ \mathit{thms})\ \mathit{thms} \\
&\qquad \texttt{else return}\ \mathit{imp})
\end{aligned}$$

The function `is_imp` returns true if its argument is an implication, while `MATCH_MP` *imp* is a function which attempts to match the antecedent of *imp* with its argument. Thus, `chain1` applies a rule of the form $P \implies Q$ across a discoverer of antecedents. The function `chain2` applies a rule of the form $P \implies Q \implies R$ across two discoverers of antecedents. The function `chain3` applies a rule of the form $P \implies Q \implies R \implies S$ across three discoverers of antecedents. The final, more general function, *recursively* applies rules with arbitrary numbers of curried antecedents from the discoverer `imps` across all possible combinations of theorems from the discoverer *xs*.

Note that the discoverers `chain1`, `chain2`, `chain3` will not necessarily try all combinations of theorems from their arguments. They fail opportunistically, attempting Modus Ponens on each theorem from the first argument, and *only if it succeeds*, attempting Modus Ponens on theorems from the second argument. This is a happy feature of the data-driven semantics of monads (but see §3.9 for its drawbacks).

It is therefore sensible to order the antecedents of the implication according to how likely they are to succeed. Antecendents which rarely succeed should appear early to guarantee a quick failure.

3.7 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 the 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.

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

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 theorem's justification.

The use of a function f of type `discoverer` $\alpha \to$ `discoverer` α allows us to easily give a discovery pipeline via function composition to justify a declarative step. To give an example, later on 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 and, finally, we have a function `add_triangles` which introduces non-collinearity claims from theorems 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 keyword `clearly`. This has the same type as `obviously`, but rather than collecting all theorems generated by a discoverer, it searches specifically for the theorem that the user is trying to justify in the current step (in the case of a `qed` step, this is just the goal theorem). The keyword will not help in proofs unless the user knows the form of theorems produced by the discoverer, but when it can be used, it leaves the basic Mizar light justification tactic with a trivial proof.

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

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

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 facts are then supplied to the function *tac* (often just `MESON`).

3.7.1 Concurrency

In §3.3.1, we described how we could easily exploit concurrency with our näive forward-chaining algorithm. Things are even better for our discovery algebra. Since we are using lazy lists pervasively, theorems can be output to the terminal one-by-one as they are generated, without having to wait for an entire generation at once.

It is also trivial to define a new primitive discoverer `monitor` which can monitor the proof-state. As we stated in our introduction in §3.4.3, 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`.

To complete the architecture, we supply basic commands that handle the thread communication necessary to tell an existing concurrent discoverer to pause, resume, or to change to an alternative theorem discoverer.

3.7.2 Dependency Tracking

While writing a proof, we would normally start an incidence discoverer concurrently, which could generate theorems as we worked. The discoverer would consider *all* of our goal hypotheses, and would typically find a large number of theorems. Most of these theorems are not needed in the proof. Of those which are needed, we would like to know the specific subset of hypotheses from which they are 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 in a computation, and for running computations to retrieve the final written values. It can be seen as 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 easily 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 hypotheses sets with the union operation. These sets contain the dependent hypotheses for every discovered theorem, and they automatically accumulate as theorems 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 do is extend the `monitor` discoverer to initialise the dependent hypotheses sets. That is, every time `monitor` pulls a hypothesis from the goal context, it must also *write* the hypothesis as a dependency on further computation. The dependencies then automatically propagate.

3.8 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 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 Spivey's original 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. [3]

3.8.1 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 uppercase 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

[3]Our code, including our revisions to Ocaml's defacto lazy list library is available on GitHub at `https://github.com/Chattered`.

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.

We have stressed "concurrency" in this chapter, rather than "parallelisation", since Ocaml only supports the former. This means we get no speed improvements by using threads. This should not be seen as a significant drawback, since such multithreading is inherently dangerous in a language such as Ocaml which allows for uncontrolled side-effects. HOL Light itself was not developed with multiprocessing support in mind. Its basic `MESON` tactic, for instance, is not thread-safe. Even without parallelisation, the tactic must be avoided when implementing discoverers.

Another issue with threading concerns UNIX signals. The standard way to interact 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` [4]. 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 and simply discard it.

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 extremely cumbersome. Primitive list functions have to explicitly force lazy constructors, 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. These actually arise quite easily when writing discoverers in our framework, and are fixed with choice uses of the `delay` function.

[4] I believe Lucas Dixon once described these procedures as CTRL-C complete.

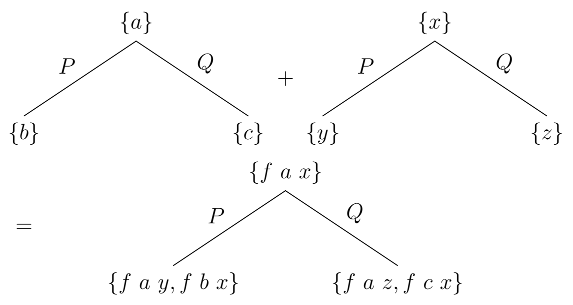


Figure 3.10: Applicative Simplification

3.9 Applicative Functors

An idiom or applicative functor [44] F is a generalisation of the monad, which could be thought of as providing at least the ability to lift values of type $\alpha \to F\ \alpha$ 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 [64] 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 a performance issue with our trees.

In Figure 3.10, 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 values in the two trees. The results were discarded in simplification, but only after they were evaluated. This completely breaks the intended purpose of the trees, which is to partition the values 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 3.10, 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 overrided 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.

3.10 The Problem Revisited

We now return to our original data-flow problem. In Figure 3.11, we use our stream algebra to capture the complex data-flow network from Figure 3.1. The five kinds of theorem now correspond to five primitive discoverers. Inference rules are applied across the theorems by lifting them into the type of discoverers, and mutual dependencies of the network are captured by mutual recursion[5].

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 3.1 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`

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

```
    ssum = foldr (+) 0 ∘ map return


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 (fmap′ 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 3.11: Incidence Discovery

into two discoverers, `neqs` and `neqs'`.

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

3.11 Conclusion

In this chapter, we began by looking for a way to implement an automated incidence tool, suitable for use with declarative proof. We began with a näive approach, and implemented a simple forward-chaining algorithm. While very basic, the advantage of this algorithm was that it makes it almost trivial to implement a collaborative and concurrent proof system, where the user and theorem prover progress in the same direction towards the goal by sharing theorems.

To correct the various deficiencies and the ad-hoc nature of our implementation, we implemented a discovery algebra based on Spivey's stream monad. The advantages of using monads is that, firstly, they are a well-established pattern in functional programming, and secondly, they use combinators so that the defined languages integrate easily with the host language.

We showed how to integrate our algebra into the rest of HOL Light, providing interfaces to the basic rewriter, the tactics and the Mizar Light combinators. Finally, we showed how to express our complex data-flow for forward-chaining incidence reasoning using our combinators in what we hope is a reasonably clear manner.

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

3.12 Further Work

The theoretical underpinnings of our tree data structure still need some investigation. The correctness of our simplification algorithm needs to be explored and ideally formally verified. For now, we regard this as beyond the scope of our current work. We still need to devise a way to integrate proper subsumption into our algebra as mentioned in §3.6.3. 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, and so it is perhaps premature to start working on proofs for our algebra. The most that we can say for now is that our incidence reasoner has been extensively tested on non-trivial problems.

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.

Chapter 4

Elementary Consequences in Group II

In this chapter, we look at the first three theorems of Group II, which happen to be the only theorems in the group which have prose proofs. Each proof makes extensive use of Axiom II, 4. In Chapter 2, we explained how this axiom had been particularly difficult to apply in our earlier work [65] and the work of Meikle and Fleuriot [48]. Each time the axiom is applied, many preconditions about incident points must be established, and we usually have to discard a disjunct in the conclusion, again appealing to incidence reasoning.

In the last chapter, we described some automation to handle this complex incidence reasoning. An aim of the present chapter is to see how much this automation can help us explore proofs, and help us shorten our verifications and make them easier to develop.

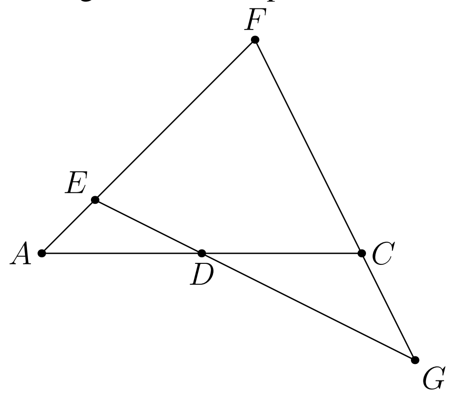
For an ancillary aim, we consider to what extent computer-aided verification, supplemented with appropriate domain-specific automation, can be used as an *analytical tool* with which to inspect informal arguments. For this chapter only, we shall be looking at Hilbert's prose proofs at a very fine detail. With the aid of machine-checked and faithful verifications of the prose, we can reason at this level of detail with confidence.

4.1 Theorem 3

Hilbert's first theorem tells us that there is a point between any two others, or as Pasch would have phrased it, that every segment contains a point.

> 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.



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. This is consistent with a claim we made in Chapter 2: Hilbert mostly appeals to axioms which *introduce* points, because the most important steps of synthetic geometry are those which obtain our diagrams. Here, we have Axiom I, 3 which can be used (indirectly) to obtain points off of arbitrary lines, in this case the point F. We have Axiom II, 2 which is our "line-extension axiom", used here to obtain the point F, and 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 an axiom in the first edition of the *Foundations of Geometry*, and the redundancy might still come as a surprise. We are obtaining an entirely linear theorem from a simple one-dimensional order axiom (II, 2) and the two-dimensional Pasch 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.

4.1.1 Verification

Our verification is shown in Figure 4.1. Our earlier declarative Isabelle verification [65] 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.

We also cite two extra axioms. We mentioned that 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 incidence theorems considered in the last chapter into a single discoverer, while the latter two just discover theorems of particular kinds. The semantics of laziness takes care of making this a genuine optimisation: if we only pull theorems from the `by_neqs` discoverer, no theorems 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 3.1).

Finally, we have used `MATCH_MP` to apply a version the Pasch Axiom (B.3) phrased in terms of point sets. This is just there to help the default prover at this step. We do not break the declarative style, since `MATCH_MP` has no side-effects on the proof context, and we still explicitly mention the 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 §4.2.1.

4.1.2 The Outer and Inner Pasch Axioms

As we said earlier, Theorem 3 was an axiom of the first edition of *Foundations of Geometry*. The full investigation of these redundancies 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 work [77], we see he chose a different rendering of the Pasch 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.*

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 an important advantage 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 [58]), 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 formalised as:

$$\neg(\exists a. \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ B$$
$$\implies \exists F a. \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F \wedge \texttt{between}\ A\ F\ C \quad (4.1)$$

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 derive Inner Pasch as a theorem and use it as an alternative to Axiom II, 4, we get a verification which does not need the final step we had before. In fact, had we spotted the factoring in our Isabelle formalisation, we predict that we would have only needed an eight rather than twenty-two step Isabelle verification.

`assume` $A \neq C$

`so consider` E `such that` $\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ E\ a)$

  `by` (I, 2), (I, 3.2) 0

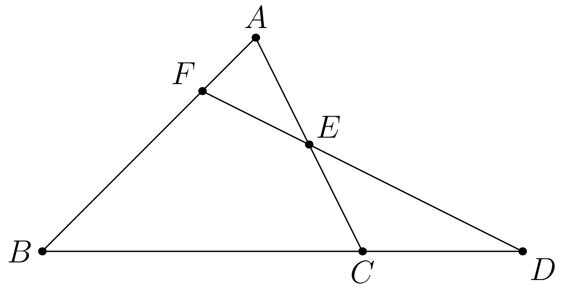
`obviously` `by_neqs` `consider` F `such that` `between` $A\ E\ F$ `from` 0 `by` (II, 2) 1

`obviously` `by_neqs` `so consider` G `such that` `between` $F\ C\ G$ `from` 0 `by` (II, 2) 2

`obviously by_ncols so consider` D `qed from` 0, 1 `by` (II, 1), (4.1)

For comparison, here is Veblen's proof of the same theorem. The two are exactly analogous. Veblen's diagram just replaces the Inner Pasch axiom with the Outer Pasch axiom:

> TH 6 [...]. Between every two distinct points there is a third point.
> Proof. Let A and B be the given points [figure]. By theorem 5 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.



In conclusion, if we judge Hilbert's argument for Theorem 3 to be gappy because of missing incidence reasoning, then we can bring it up to the more pedantic standards of Veblen. Before Theorem 3, we suggest one first derive the Inner Pasch axiom (4.1), after which the proof of Theorem 3 follows more fluidly. This observation will be worth bearing in mind when we come to Theorem 5 in §4.3.

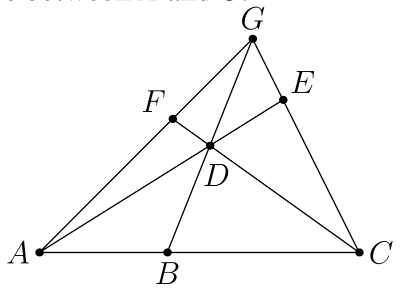
4.2 Theorem 4

In this section, we shall review more carefully how we can use our discovery tool in an exploratory fashion, as we examine Hilbert's Theorem 4. In the first edition of *Foundations of Geometry*, Hilbert had incorporated this theorem 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.*

The "only one" direction is all that is retained in the Tenth Edition. The other direction was 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 [(Pasch's axiom)] 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 realizes that B lies between A and C.



4.2.1 Discovering Applications of Pasch

With Axiom II, 4 used a total of four times in this theorem, and with the symmetry that is obvious in the diagram, we wanted to explore this proof a bit more using our automated discoverers. One drawback of our discoverers is that, even though they tell us various incidence relations implied by our assumptions, and thus help us discharge preconditions on Axiom II, 4, they do not tell us which instantiations of this axiom are possible.

We cannot simply define a new discoverer which finds possible applications of Axiom II, 4 by using our simple forward-chaining primitives `chain1`, `chain2`, `chain3`. The problem is that our version of this axiom in point sets (2.13) 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 theorems 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.

This discoverer outputs theorems which are the conclusions of Axiom II, 4, which therefore tell us where Axiom II, 4 can be applied. At the point in Hilbert's proof where Axiom II, 4 is first used, the discoverer only finds two possibilities. Hilbert uses both of them.

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

After Hilbert applies the first of these, another three possibilities arise, depicted in Figure 4.2

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

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

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

Cases (a) and (c) of Figure 4.2 obtain the exact same point F, but in case (a) we should `assume` $A \neq C$

`so consider` E `such that` $\neg(\exists a.$`on_line` $A\ a \wedge$ `on_line` $C\ a \wedge$ `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`

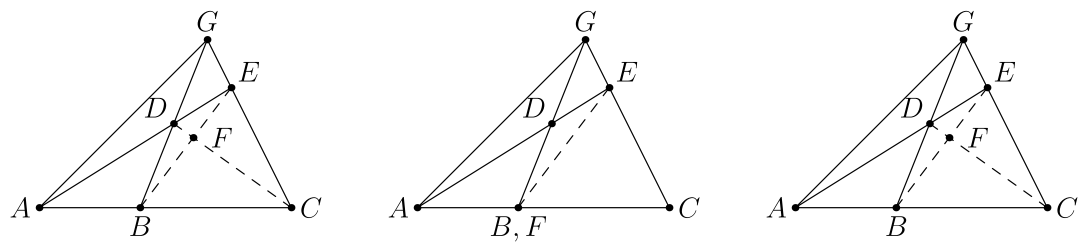
  $(\exists a.$`on_line` $E\ a \wedge$ `on_line` $G\ a \wedge$ `on_line` $D\ a)$

  $\wedge$ (`between` $A\ D\ C \vee$ `between` $F\ D\ C$)

  `using K (MATCH_MP_TAC` (B.3)`)` `from` 0, 1

`obviously` (`by_eqs` $\circ$ `split`) `qed from` 0, 1, 2 `by` (II, 1), (II, 3)

Figure 4.1: Verification of Theorem 3



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

Figure 4.2: Three Possible Applications of Axiom II, 4

conclude that F is between B and E while in (b) 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 §4.2.3.

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 discoverer, cannot reject the offending disjunct in the axiom's conclusion.

4.2.2 Verifying Hilbert's Proof

We verified this theorem declaratively in Isabelle, where it ran to sixty-nine steps. In such a long proof, the structure of the basic prose argument is buried by incidence arguments which do not illuminate anything. They are mostly applications of our point set rules given in §2.4 with manual variable instantiations. We tried to use comments to show how the prose translated into the formal proof steps, but in the end, the hope of using verification *analytically* had been lost.

But with our incidence automation, we have a HOL Light proof in just thirteen lines, each readily understandable, and matching the prose very closely. In fact, in earlier work [66], we matched the prose one-one, but only by using our much more ad-hoc automation based on our naïve approach from §3.3.

In our current verification shown in Figure 4.3, we limit the automation to the `by_pasch` discoverer, and are satisfied that we are close enough to Hilbert's prose. Unlike the more ad-hoc automation, this discoverer will not automatically discharge case-splits in the conclusion of Axiom II, 4. The elimination of the disjunct, and any identification of the point obtained with an existing point, is performed using the `by_eqs` discoverer in a subproof.

At the start of the verification, we have set our ultimate goal conclusion to be

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

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.

```
assume $\exists a.$on_line $A\ a \wedge$ on_line $B\ a \wedge$ on_line $C\ a$                      0
assume $A \neq B \wedge A \neq C \wedge B \neq C$                                                  1,2,3
assume $\neg$between $A\ C\ B \wedge \neg$between $B\ A\ C$                                        4
consider $D$ such that $\neg(\exists a.$on_line $A\ a \wedge$ on_line $B\ a \wedge$ 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 $(\exists a.$on_line $A\ a \wedge$ on_line $D\ a \wedge$ on_line $E\ a)$    7
                                    $\wedge$ between $C\ E\ G$ proof                               8
    clearly by_pasch consider $E$ such that
        $(\exists a.$on_line $A\ a \wedge$ on_line $D\ a \wedge$ on_line $E\ a)$
        $\wedge$ (between $B\ E\ C \vee$ between $C\ E\ G)$ by (II, 1) from 0,2,3,5,6
    obviously (by_eqs $\circ$ split) qed from 0,3,4,5 by (II, 1),(II, 3)
consider $F$ such that $(\exists a.$on_line $C\ a \wedge$ on_line $D\ a \wedge$ on_line $F\ a)$    9
    [...]
have between $A\ D\ E$ proof
    obviously by_ncols so consider $D'$ such that
        between $C\ D'\ F \wedge$ between $E\ D'\ A$
        using K (MATCH_MP_TAC (4.4)) from 0,2,5,6,8,10 by (II, 1)
    obviously (by_eqs $\circ$ split) qed from 0,2,5,7,9 by (II, 1),(II, 3)
obviously by_ncols so consider $B'$ such that
    between $G\ D\ B' \wedge$ between $C\ B'\ A$
    using K (MATCH_MP_TAC (4.3)) from 0,2,5,7, by (II, 1)
obviously (by_eqs $\circ$ split) qed from 0,2,5,6 by (II, 1),(II, 3)
```

Figure 4.3: Verification of Theorem 4

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 conclusion becomes just

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

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 having to eliminate a disjunct in the conclusion of Axiom II, 4. To do this, we find point equalities via the composed discoverer $\texttt{by\_eqs} \circ \texttt{split}$, 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.

In these two applications of Axiom II, 4, we have used our `clearly` keyword to search for a specific conclusion of the axiom. Exploiting our discoverer in this way makes things more robust than if we use `MATCH_MP_TAC` as we did in §4.1. With matching, we have to be careful that all the variables in the axiom line up with our desired conclusion. 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. Besides, we usually just copy and pasted the conclusion directly from the results of a concurrent run of the discoverer.

The third and fourth applications of the Pasch axiom can take the inner (4.4) and outer (4.3) forms respectively. This gives us our second recommendation for a useful lemma to insert in Hilbert's theory: we should prove the Outer Pasch axiom.

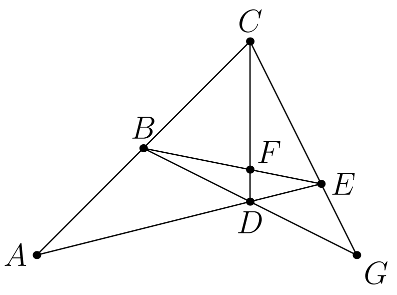
4.2.3 Alternative Proof

When we were still using our more ad-hoc discovery tool, we decided to try applying Axiom II, 4 non-deterministically. The purpose of the tool was to search for alternative proofs of Theorem 4, and hopefully find one which required less than four applications of the axiom. This defined a search limit for the problem: once four applications were found, it stopped searching in the relevant branch.

The tool found several "alternatives", but most of these yield symmetries of the original proof. In some cases, two independent applications of Pasch's axioms were applied in reverse order. In other cases, the proof was identical to the original up to a symmetric relabelling of the points involved in the diagram. Only one new proof was revealed up to symmetry, and it corresponds to case (a) and (c) of Figure 4.2. We give it now in a prose formulation with an accompanying diagram.

Theorem. *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 it 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. □



In some sense, the basic strategy of our alternative proof is the same as the original. We first construct a point D off the line AC and extend the line 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 a Pasch 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 it is clear that the third application of Pasch 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 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 Theorems 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*. The reasoning may be laborious, but as we have seen, the situation can sometimes be improved by first proving stronger variants of Axiom II, 4.

4.3 Theorem 5

For the final theorem we shall consider in this chapter, we shall see once again how much we gain when we think in terms of the inner and outer form of the Pasch axiom, and we will look under the hood to see what our discoverers are actually up to.

Theorem 5 is the most complex of the three theorems, taking up almost an entire page of the English translation. Like the proceeding two theorems, 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 theorem gives a transitivity property for point ordering. The proof is divided into three parts, though as observed by Dehlinger et al [12], it makes sense to leave the third part to the generalisation of Theorem 6. We leave this until the next chapter.

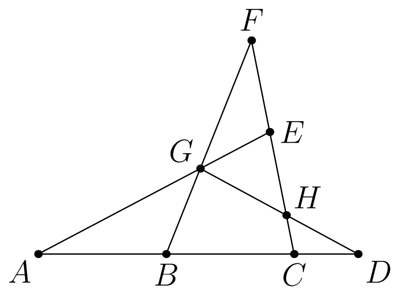
4.3.1 Part 1 of Theorem 5

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 labeled B lies between A and C and also between A and D, and furthermore, that the point labeled

C lies between A ad 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, [and] 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.



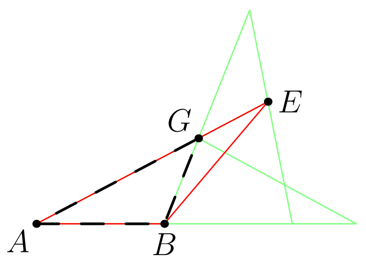
4.3.1.1 Our Isabelle Verification

Our Isabelle verification of this proof runs to approximately 135 lines of often complicated formal proof steps. As we should expect by now, most of these steps were used to derive the preconditions needed for Axiom II, 4, all of which have been eliminated in the HOL Light verification.

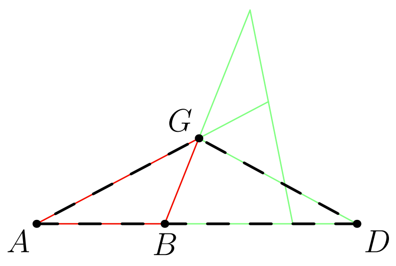
In Isabelle, the complexity of the inferences had got the better of us. We were not able to verify Hilbert's final application of Axiom II, 4 with the line EH and the triangle ADG. This requires knowing that the line EH does not intersect any vertex of A, D, G, which requires, in particular, knowing 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 (thankfully, the derivation turned out to be invalid), and we hoped that our discoverers would help confirm this.

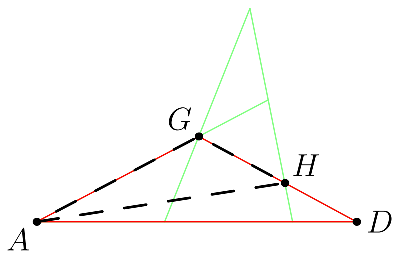
Instead, our discoverers refuted it. By tracking the path of inferences via a writer (see §3.7.2), we found that $\triangle AEH$ is derived at the end of a chain of discovered triangles starting from $\triangle ABE$. The rule linking each is `colncolncol` from §3.2, which can be understood as *substituting* points of a non-collinear triple one-at-a-time, until we have



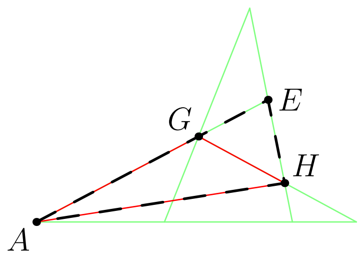
(a) $\triangle ADG$ from $\triangle ABG$



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



(c) $\triangle ADG$ from $\triangle ABG$



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

Figure 4.4: Finding $\triangle AEH$

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. The inference is less direct, and is shown in Figure 4.4. 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 $\triangle AHG$. Finally, E and H are shown distinct on the basis of $\triangle AGH$ and the line AGE, after which the discoverer substitute H for G using the line AGE, thus obtaining $\triangle AEH$.

4.3.1.2 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

$$\neg(\exists a.\texttt{on\_line}\,A\;a \wedge \texttt{on\_line}\,B\;a \wedge \texttt{on\_line}\,C\;a) \wedge \texttt{between}\,B\;C\;D \wedge \texttt{between}\,A\;E\;C \implies \exists F a.\texttt{on\_line}\,D\;a \wedge \texttt{on\_line}\,E\;a \wedge \texttt{on\_line}\,F \wedge \texttt{between}\,A\;F\;B \quad (4.2)$$

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 §4.1.2). This is an important corollary, as evidenced by the fact that it is the very first theorem Veblen proves after stating the axiom. Formally, both inner and outer Pasch axioms can be derived in strengthened forms:

$$\neg(\exists a.\texttt{on\_line}\,A\;a \wedge \texttt{on\_line}\,B\;a \wedge \texttt{on\_line}\,C\;a) \wedge \texttt{between}\,B\;C\;D \wedge \texttt{between}\,A\;E\;C \implies \exists F.\texttt{between}\,D\;E\;F \wedge \texttt{between}\,A\;F\;B \quad (4.3)$$

$$\neg(\exists a.\texttt{on\_line}\,A\;a \wedge \texttt{on\_line}\,B\;a \wedge \texttt{on\_line}\,C\;a) \wedge \texttt{between}\,B\;C\;D \wedge \texttt{between}\,A\;E\;B \implies \exists F.\texttt{between}\,D\;F\;E \wedge \texttt{between}\,A\;F\;C. \quad (4.4)$$

4.3.1.3 Verification

With these new axioms and some of our automation, we obtain the formalisation in Figure 4.5, 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 a Pasch axiom.

The other extra step is somewhat of an irritation. Instead of applying our Pasch axioms 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.

4.3.1.4 Comparison with the Prose

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

> 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 at the point GD at a point H.

In our other verification of Theorem 5, which avoids the Inner and Outer Pasch axioms and 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 $\exists a.$on_line $B\ a \wedge$ on_line $F\ a \wedge$ on_line $G\ a \wedge$
                    $\wedge$ between $A\ G\ E$ proof                        4,5
    clearly by_pasch ...
have between $B\ G\ F$ proof                                                 6
    clearly by_pasch ...
consider $H$ such that $\exists a.$on_line $C\ a \wedge$ on_line $F\ a \wedge$ on_line $H\ a$
                    $\wedge$ between $D\ H\ G$ proof                        7,8
    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 only need only one fact of betweenness.

In our verification using the Inner and Outer Pasch axioms (Figure 4.5), we can trim this down. The intersections of the segments AE and BF is covered by just one application of the strengthened version of the Inner Pasch Axiom (4.4), which we use again to intersect the segments CF and GD. Finally, we use the Outer Pasch Axiom (4.3) to find a point C' between A and D.

For this final application of a Pasch 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. Here, 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 Inner Pasch (4.4).

Finally, we elide Hilbert's final step "one shows analogously that B also lies on this segment." The analogue is just a corollary obtained by symmetry of betweenness: we just swap A and D, and swap B and C, and obtain the required result.

4.3.2 Discovered Theorems

In this subsection, we give an idea of how our discoverer interacts with HOL Light by showing the theorems generated concurrently as we develop a verification of Theorem 5. We begin with the theorem's assumptions, and then obtain our first non-collinear point.

```
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
```

Concurrently, results are being pulled from the discoverer `by_incidence`, which will lazily force values according to our data-flow diagram from Figure 3.1. This ultimately means pulling theorems from the discoverer `monitor`, whose job it is to inspect the proof context and add any new theorems which appear there. Here, our two new hypotheses will be picked up and fed through our incidence reasoner to produce seven generations of incidence theorems. If the stream is forced beyond this, only empty generations will appear, indicating that no more inference is possible.

These seven generations of theorems are delivered within 0.31 seconds. We include the dependent hypotheses in parentheses. [1]

[1]We have tested this on an Intel Core 2 2.53GHz machine.

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

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

$$
\{\},
$$

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

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

$$
\{\},
$$

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

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 facts which have already appeared, and duplicates are always removed from the inference process. This filtering sometimes leaves generations completely empty.

Besides the selection of triangles here, we have a theorem which says that all points in our figure lie in the same plane. We can see how this theorem has grown over the generations, with larger and larger planar sets found, via rule `planeplane` from §3.2.

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

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

Since this was the *last* hypothesis to enter the proof context, we can add it as justification using the Mizar Light keyword so.

$$\texttt{obviously by\_neqs so consider } F \texttt{ such that between } C\ E\ F \texttt{ from } 0 \texttt{ by } (\mathrm{II}, 2) \quad 3$$

The next sequence of generations is found within 1.21 seconds:

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

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

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

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

$$\{\},$$

$$\left\{\begin{array}{ll}B \neq F & (0,1,2,3),\\ A \neq F & (0,2,3)\end{array}\right\},$$

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

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

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

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

With thirteen triangles identified, we are probably interested in what applications of Axiom II, 4 are permissible, so we will want to look more specifically at the results of our by_pasch discoverer. Already, it is telling us that there are six possibilities to choose from. The full set is found within 2.82 seconds.

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

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

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

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

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

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

The last possibility here corresponds to one of Hilbert's intended applications of Axiom II, 4. To apply it, all we need to do is assert the theorem as `clearly` derivable, being sure to supply all four hypotheses as justification:

```
clearly by_pasch 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, 1, 2, 3
```

We now have a disjunction in our hypotheses. Provided our discoverer was called with the `split` function, it will convert this disjunction into a tree, which will combine as described in §3.5.1. Our generations will become proper trees, and the inference process will be 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. For brevity, we omit discovered theorems which are later subsumed.

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

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.

We get more information in the next partition, which is the left-branch of the case-split. When the trees are flattened, the theorems in this branch carry `between` $A\ G\ E$ as a disjunctive hypothesis.

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

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` $C\ G\ E$.

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

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

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

$\{\exists a.\texttt{on\_line}\, C\, a \wedge \texttt{on\_line}\, E\, a \wedge \texttt{on\_line}\, F\, a \wedge \texttt{on\_line}\, G\, a, \quad (0,2,3,4)\}$

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

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

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

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

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

$\{F = G \quad (0,1,2,3,4)\}$

The use of our tree data structure means we can think of this last sequence as having been 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 actually 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` $C\ E\ F$, 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 theorems as antecedents, and then uses the resulting discovered theorems 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 earlier Isabelle verification to a verification with just 17 steps. We found this roughly 80% reduction in proof length across all 18 theorems from our Isabelle verifications, with the new verifications always comparing much more favourably with the prose.

4.3.3 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.

Our verification is shown in Figure 4.6. In the prose, Hilbert is applying Axiom II, 4 in its contrapositive form, so we cannot follow Hilbert 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 makes a similar citation in a proof supplementing the text (see §6.2.2). 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 $A\ B\ C$ $\wedge$ 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)                                             $2$
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 $A\ G\ E$ $\wedge$ between $B\ G\ F$ from $0, 2, 3$ by (II, 1), (4.4)      $4, 5$
obviously by_ncols so consider $H$ such that
    between $C\ H\ F$ $\wedge$ between $D\ H\ G$ from $0, 1, 2, 3, 5$ by (II, 1), (4.4)    $6$
have $A \neq D$ from $0, 1$ by (II, 1), (II, 3)
obviously by_ncols so consider $C'$ such that
    between $E\ H\ C'$ $\wedge$ between $A\ C'\ D$ from $0, 1, 2, 4, 6, 7$ by (4.3), (II, 1)
obviously (by_eqs $\circ$ split) qed from $0, 1, 2, 3, 6, 7$
```

Figure 4.5: Theorem 5 Verification, Part 1

```
assume between $A\ B\ C$ $\wedge$ between $A\ C\ D$                                           $0, 1$
consider $G$ such that
    $\neg(\exists a.$on_line $A\ a$ $\wedge$ on_line $B\ a$ $\wedge$ 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 $\neg(\exists P.(\exists a.$on_line $C\ a$ $\wedge$ on_line $F\ a$ $\wedge$ on_line $P\ a)$ $\wedge$ between $A\ P\ G$ proof    $4$
    otherwise consider $P$ such that
        $\exists P.(\exists a.$on_line $C\ a$ $\wedge$ on_line $F\ a$ $\wedge$ on_line $P\ a)$ $\wedge$ between $A\ P\ G$    $4, 5$
    clearly by_pasch so consider $Q$ such that
        $(\exists a.$on_line $C\ a$ $\wedge$ on_line $P\ a$ $\wedge$ on_line $Q\ a)$
            (between $A\ Q\ B$ $\vee$ between $B\ Q\ G$) from $0, 2$
    obviously (by_eqs $\circ$ split) qed from $0, 2, 3, 4, 5$ by (II, 1), (II, 3)
obviously by_pasch so consider $H$ such that
    $(\exists a.$on_line $C\ a$ $\wedge$ on_line $F\ a$ $\wedge$ on_line $H\ a)$
        $\wedge$ between $D\ H\ G$ from $0, 1, 2, 3$ at $5, 6$
have $B \neq D$ from $0, 1$ by (II, 1), (II, 3)                                              $7$
clearly by_pasch so consider $C'$ such that
    $(\exists a.$on_line $C'\ a$ $\wedge$ on_line $F\ a$ $\wedge$ on_line $H\ a)$
        (between $B\ C'\ D$ $\vee$ between $B\ C'\ G$ from $0, 1, 2, 3, 5, 6$
obviously (by_eqs $\circ$ split) qed from $0, 1, 2, 3, 5, 6, 7$ by (II, 1), (II, 3)
```

Figure 4.6: Theorem 5 Verification, Part 2

4.4 Conclusion

Hilbert gives only three prose proofs for the first two groups of axioms in *Foundations of Geometry*. None of these proofs were present in the first edition, and the theorems they prove, one-dimensional ordering theorems, were assumed as axioms in the first edition. 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 Isabelle 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 incidence discoverers appear to be adequate to handle all of the incidence reasoning implicit in Hilbert's proofs.

As we move onwards from these simple theorems, we will see how much we came to rely on our incidence automation. As we introduce new concepts such as triangle interiors and exteriors in Chapter 9, we will see that it is prudent to add component discoverers which can then be composed with the incidence discoverers from the last chapter. The composed discoverers are used pervasively in Chapters 9 and 10.

We would eventually like to tackle Hilbert's Group III, which contains many new prose proofs. We are confident, based on our experience verifying these theorems in earlier work [65], that our incidence discoverers will make short work of Hilbert's implicit argumentation in these proofs.

Finally, we have shown how Hilbert's proofs can be substantially improved upon if we derive Inner and Outer Pasch versions of Axiom II, 4. These exchange most of the complex preconditions of Pasch's axiom, including the tedious assumption that all points considered by the axiom lie in a plane, for just one additional betweenness assumption.

Chapter 5

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 are linearly ordered in terms of betweenness. One approach to formalising this is at the metalevel, 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. As mentioned in §1.3.2, this axiom is redundant given Hilbert's geometric axioms.

5.1 Theorem 6 at the Metalevel

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

> THEOREM 6 (generalization 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.

A *labelling* of a point is, of course, a syntactic device, *used* to *mention* points. This distinction was known to logicians of the time, and probably to Hilbert, even though he was working before it had been formalised.

Thinking in terms of formal languages, we could identify a *labelling* as the assignment of an existing value to a new variable. The existence of a labelling is then the existence of these assignments. With this in mind, we could semi-formalise Theorem 6 to say that, given some points on a line A', B', C', D', ..., K', there exists an appropriate assignment to A, B, C, D, ..., K:

$$
\begin{aligned}
&\exists a.\mathtt{on\_line}\ A'\ a \wedge \mathtt{on\_line}\ B'\ a \wedge \mathtt{on\_line}\ C'\ a \wedge \mathtt{on\_line}\ D'\ a \\
&\qquad \wedge \mathtt{on\_line}\ E'\ a \wedge \cdots \wedge \mathtt{on\_line}\ K'\ a \\
&\Longrightarrow \exists A\ B\ C\ D\ \ldots\ K. \\
&\qquad\quad A = A' \wedge B = B' \wedge C = C' \wedge D = D' \wedge E = E' \wedge \cdots \wedge K = K' \\
&\qquad \wedge (\mathtt{between}\ A\ B\ C \wedge \mathtt{between}\ A\ B\ D \wedge \mathtt{between}\ A\ B\ E \wedge \cdots \wedge \mathtt{between}\ A\ B\ K) \\
&\qquad \wedge \begin{pmatrix} \mathtt{between}\ A\ C\ D \wedge \mathtt{between}\ A\ C\ E \wedge \cdots \wedge \mathtt{between}\ A\ C\ K \\ \wedge \mathtt{between}\ B\ C\ D \wedge \mathtt{between}\ B\ C\ E \wedge \mathtt{between}\ B\ C\ K \\ \wedge \cdots \wedge \mathtt{between}\ B\ C\ K \end{pmatrix} \\
&\qquad \wedge \begin{pmatrix} \mathtt{between}\ A\ D\ E \wedge \cdots \wedge \mathtt{between}\ A\ D\ K \\ \wedge \mathtt{between}\ B\ D\ E \wedge \cdots \wedge \mathtt{between}\ B\ D\ K \\ \wedge \mathtt{between}\ C\ D\ E \wedge \cdots \wedge \mathtt{between}\ B\ D\ K \end{pmatrix} \\
&\qquad \wedge \mathtt{between} \ldots .
\end{aligned}
\tag{5.1}
$$

Obviously, we cannot capture this directly in first-order logic. The fact that we quantify over an unspecified number of variables already renders that goal hopeless. Instead, the use of ellipses here tells us that this is a *scheme*. As we fix our choice of variables in $A, B, C, D, E, \ldots, K$, we are expected to fill in the holes by continuing a syntactic pattern.

In short, this is a metatheorem, something perhaps surprisingly typical of mathematics. As noted by Harrison [22], 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 metalevel and object level, let alone the sort of hard distinction we now have between HOL and the programming metalanguage (ML) which encodes its syntax and inference rules. With such a distinction, we immediately realise that our schematic above can be formalised as an ML function which inputs a list of points, fills in the scheme to create a true HOL term, and then derives it as a theorem.

5.1.1 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 theorem we must generate is then:

$$\begin{aligned}
&\exists a. \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \\
&\implies \exists A\ B\ C\ D.\ A = A' \wedge B = B' \wedge C = C' \wedge D = D' \\
&\qquad \wedge \texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ B\ D \\
&\qquad \wedge \texttt{between}\ B\ C\ D
\end{aligned} \tag{5.2}$$

Now we can unwind this existential into the disjunction given in Figure 5.1, and thus this sufficies as a 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 this 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.

Typically, a geometry will have a specified origin and a specified axis on which to define the preferred orientation. Even coordinate free methods such as the signed-area method [32] specify a preferred orientation. Not so in Hilbert's geometry. But we *can* treat the first argument to the between predicate as a parameter giving the preferred origin and direction, and then regard the partially applied relation as a two place binary order. In other words, we can read `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

$$\begin{aligned}
&\texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \wedge \texttt{between}\ A\ D\ E \\
&\qquad \wedge \texttt{between}\ A\ E\ F \wedge \texttt{between}\ A\ F\ G \wedge \texttt{between}\ A\ G\ H.
\end{aligned} \tag{5.3}$$

We want to retrieve all other between relations implied by this term quickly. Before we describe how we do this, we will interpret Theorem 5, whose verification we considered in the last chapter, as two separate rules. The first rule (5.4) will "move the origin" in a total order. The second rule (5.5) is just transitivity if we think of the

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

Figure 5.1: Theorem 5 Case Split

between relation as a parameterised binary relation.

$$\texttt{between}\,A\;B\;C \wedge \texttt{between}\,A\;C\;D \implies \texttt{between}\,B\;C\;D \tag{5.4}$$

$$\texttt{between}\,A\;B\;C \wedge \texttt{between}\,A\;C\;D \implies \texttt{between}\,A\;B\;D \tag{5.5}$$

We now explain our strategy by way of example. Suppose our goal is to derive $\texttt{between}\ C\ F\ H$ from (5.3). Our first task is to “move the origin” from C to A. To do so, we match against (5.4), to yield:

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

We now just reason transitively from the conjuncts of the ordering. We split (5.3) into the two suborderings either side of F

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

and

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

and then obtain $\texttt{between}\,A\ C\ F$ and $\texttt{between}\,A\ F\ H$ by folding Rule (5.5) 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

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

We have implemented this procedure as a completely deterministic tactic, taking a theorem such as (5.3) and a term such as $\texttt{between}\,C\ F\ H$, and then deriving the term from the order in one pass of the order conjunction.

5.1.2 Enumerating Possible Orderings

Even with the more concise representation of orderings, there are $\frac{1}{2}n!$ orderings to consider for n points. In practice, when we want to apply the theorem, there are usually additional constraints on the ordering that allow us to eliminate most 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. Thus, the procedure we have defined to enumerate all possible orderings takes both a list of the points we want it to order, and a list of betweeness constraints on those points.

This 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 metalevel down to the object level.

5.2 Theorem 6 at the Object Level

Any logic which can formalise Theorem 6 will need to include a domain of labellings, which is potentially infinite. One such domain appears in Dehlinger et al's verification [12] 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:

$$\begin{aligned}\texttt{ordering}\, f\, X \iff X = f\,(\{n \mid \texttt{finite}\, X \implies n < |X|\})\\ \wedge \forall n\, n'\, n''.\texttt{finite}\, X \implies n < |X| \wedge n' < |X| \wedge n'' < |X|\\ \wedge\, n < n' \wedge n' < n'' \wedge \texttt{between}\,(f\, n)\,(f\, n')\,(f\, n'').\end{aligned} \tag{5.6}$$

Our definition here allows for the possibility that f is an ordering of a (countably) infinite set X. In this infinite case, we just drop the constraint that n, n' and n'' are bounded by its finite cardinality.[1]

Theorem 6 can then be formalised concisely as:

$$\texttt{finite}\, X \wedge \texttt{collinear}\, X \implies \exists f.\texttt{ordering}\, f\, X. \tag{5.7}$$

5.3 Natural Numbers

Our definition assumes the existence of natural numbers, but Hilbert, writing before the investigations of the logicists, 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)". 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]

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.[16]

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

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 they 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 [39]. Instead, numbers were to be grounded on secure geometrical notions.

5.3.1 The Axiom of Infinity

The theorem stating that natural numbers exist is derivable in HOL Light assuming a domain exists which is Dedekind-infinite.

$$\exists f : ind \to ind.\mathtt{one\_one}\ f \land \neg\mathtt{onto}\ f. \tag{5.8}$$

This theorem asserts that there is a one-one but not onto function f. 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 hypothesis on the theorems which required it [56]. Without the axiom, one must allow for the possibility that our 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." Thus, we should be able to derive Theorem 5.8 from Hilbert's geometric axioms. We can then obtain the natural numbers from this theorem, replacing the abstract type *ind* with a concrete type of geometric objects. Our natural numbers will then be quite literally founded in geometry.

We can easily capture this in HOL Light, since 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 will 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.

5.3.2 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 by writing, for appropriate l and p:

$$\texttt{Group\_I}\ l\ p$$

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

$$\texttt{new\_axiom}\ (\texttt{Group\_I on\_line on\_plane}).$$

Besides allowing us to do this, the predicate `Group_I` allows us to express basic metatheoretical ideas about the Group I axioms, and if we treat the hypothesis as a *context* of axioms, we can regard it as a basic mechanism to write a module of incidence proofs.

In general, such techniques have significant weaknesses. For one, we are limited in how much we can reason about theories of polymorphic values. We would want to be able to do this if we wanted to reason about the theory of monads in Chapter 3. This would require a stronger type-theory, such as the extension provided by HOL Omega [30].

For another, when we want to treat `Group_I on_line on_plane` as a module of theorems, we would still 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 [35].

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 reasoning in later groups such as Group III where axioms are defined based on quite complex and derived definitions.

5.3.2.1 Formalisation

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.

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

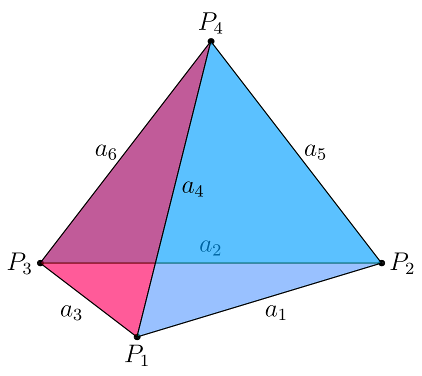
The type definitions are used to automatically derive an abstract type and derive two theorems, an induction theorem and a recursion theorem. For the type `ps`, for instance, we are given

$$\forall P.\ P\,\mathtt{p1} \wedge P\,\mathtt{p2} \wedge P\,\mathtt{p3} \wedge P\,\mathtt{p4} \implies \forall p. P\ p$$
$$\forall p1\ p2\ p3\ p4.\ \exists f.\ f\,\mathtt{p1} = p1 \wedge f\,\mathtt{p2} = p2 \wedge f\,\mathtt{p3} = p3 \wedge f\,\mathtt{p4} = p4.$$

The first (induction) theorem can be promoted to an equivalence, and then used to rewrite all universally quantified statements as finite conjunctions. Similarly, its rewrite using the infinite DeMorgan rule, $(\forall x. Px) \iff \neg\exists x.\neg(Px)$ can be used to rewrite all existentially quantified statements 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 mere tautology and then quickly verified.

```
 on_line p1 l1 ∧ on_line p2 l1
∧on_line p1 l2 ∧ on_line p3 l2
∧on_line p2 l3 ∧ on_line p3 l3
∧on_line p1 l4 ∧ on_line p4 l4
∧on_line p2 l5 ∧ on_line p4 l5
∧on_line p3 l6 ∧ on_line p4 l6
```



```
 on_plane p1 p1 ∧ on_plane p2 p1 ∧ on_plane p3 p1
∧on_plane p1 p2 ∧ on_plane p2 p2 ∧ on_plane p4 p2
∧on_plane p1 p3 ∧ on_plane p3 p3 ∧ on_plane p4 p3
∧on_plane p2 p4 ∧ on_plane p3 p4 ∧ on_plane p4 p1
```

Figure 5.2: Smallest model

To check our model, we first inductively defined the incidence predicates `on_line` and `on_plane` over the types `ps` and `lines`, as shown in Figure 5.2. We then formalised the following theorem in HOL Light

```
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 §2.2.2. 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 four points, six lines and four planes satisfying the conditions we gave to inductively define `on_line` and `on_plane` in the model. We know from Theorem 2.2.4 that there are three distinct points on a plane. From axiom `g18`, 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.

5.4 Infinity

By Group II, there are only infinite models. Indeed, would can just apply Axiom II, 2 repeatedly to obtain an arbitrary number of points. Starting from distinct points A and B, we can obtain points A, B, C, D, E, $\ldots$, Y, Z, satisfying

$$\begin{array}{l}\texttt{between}\,A\;B\;C\\ \texttt{between}\,A\;C\;D\\ \texttt{between}\,A\;D\;E\\ \ldots\\ \texttt{between}\,A\;Y\;Z\end{array}$$

With Theorem 5, we can move from theorems such as $\texttt{between}\,A\;B\;C$ and $\texttt{between}\,A\;C\;D$ to $\texttt{between}\;A\;B\;D$, and so can prove that the points above are mutually distinct. This process therefore gives us a potentially infinite number of points.

5.5 A Geometric Successor

The function which is said to exist in the theorem of infinity is a successor function. It is possible to witness this theorem, and, staying faithful to Hilbert's prose, we will use a witness that is effectively a function between any two points of a line segment. The witness is based on the diagram in Figure 5.3.

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

$$\begin{aligned}&\neg(\exists a.\texttt{on\_line}\,A\;a\wedge\texttt{on\_line}\,B\;a\wedge\texttt{on\_line}\;0\;a)\\ &\quad\wedge\texttt{between}\,A\;D\;B\wedge\texttt{between}\;B\;0\;C\\ &\quad\wedge(\texttt{between}\,A\;N\;0\vee N=0)\end{aligned}\tag{5.9}$$

Our natural numbers are carved out from the set of all diagrams satisying this property. We abstract this set into a type (`ind`), giving us two functions: a constructor `mk_ind` which promotes any diagram into the abstract type, and a destructor `dest_ind` which converts an inhabitant of the abstract type into its diagram representation. In order for

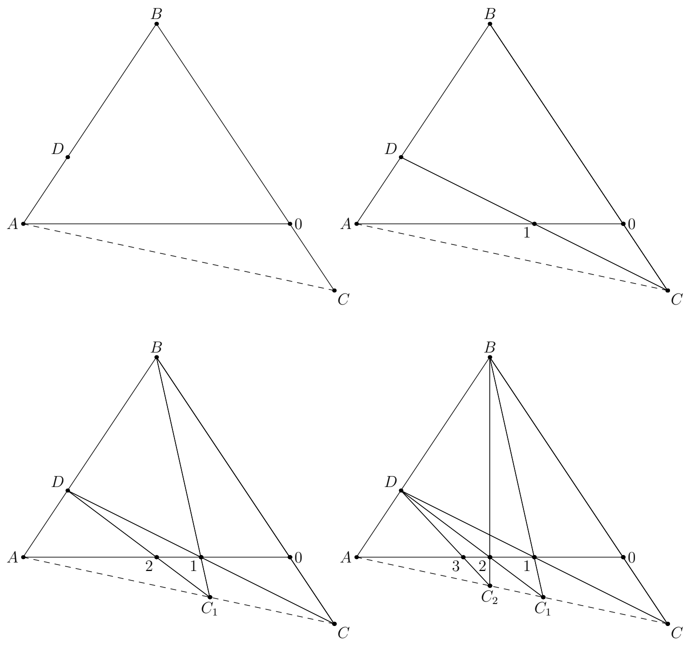


Figure 5.3: The successor function

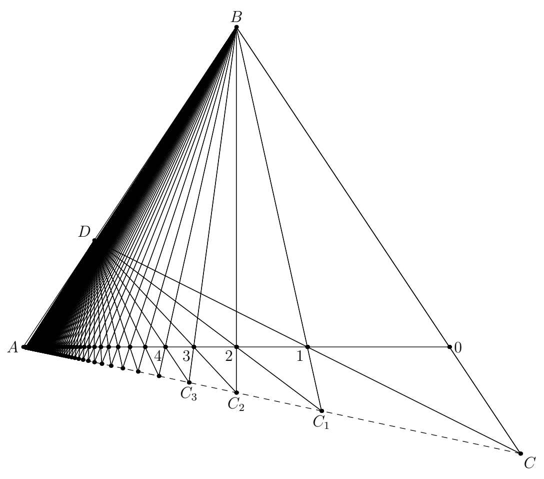


Figure 5.4: Successors tending to A

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 (§4.1).

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 5.3, 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 will identify diagrams up to the equality of the five fixed points. We can then determine a diagram from the 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 $1D$ 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 DN and AC, and then finding the intersection of $C'D$ and $A0$. Formally:

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

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 equivalent to the full axiom of choice.

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

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

$$\iota x.P\,x = \varepsilon x.P\,x \wedge \forall y.P\,y \implies x = y$$

By using this operator and avoiding the somewhat controversial 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. We need enough information in each of our figures to constrain the possible placement of successors relative to their predecessors.

5.5.1 Lemmas

Our successor function deconstructs an abstract diagram into its six points. It then 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 non-arbitrary. Formally:

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

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. In each case, we apply Pasch's axiom (II, 4) to $\triangle AB0$ and the line CD to obtain a point S between A and 0. For our initial diagram, 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 DN, to place the point C' between A and C. We can now use the previous argument to 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.

5.6 Theorem of Infinity

Finally, we must verify that our function is one-one but not onto. To verify that `ind_suc` is one-one, we went procedurally, but with the assistance of our discoverers via `discover_tac by_eqs`. This used incidence reasoning to automatically find equalities that arise from our diagrams.

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.

$$\texttt{one\_one ind\_suc} \wedge \neg \texttt{onto ind\_suc}. \tag{5.7}$$

This now shows that the abstract type `ind` has an infinite domain. However, the domain is not the natural numbers. Since we are allowed to take the successor in any diagram where the point N is any point lying between A and 0, or is the point 0 itself, it is clear that most of the images of `ind_suc` 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.

5.7 Theorem 6 Revisited

Now that we have two ways to carve out the natural numbers from geometric objects, we can return to our verification of §5.1. Recall our definition of an *ordering* from §5.2:

$$\begin{aligned} \texttt{ordering}\, f\, X \iff X = f\,(\{n \mid \texttt{finite}\, X \implies n < |X|\}) \\ \wedge \forall n\, n'\, n''. \texttt{finite}\, X \implies n < |X| \wedge n' < |X| \wedge n'' < |X| \\ \wedge n < n' \wedge n' < n'' \wedge \texttt{between}\,(f\, n)\,(f\, n')\,(f\, n''). \end{aligned} \tag{5.6}$$

There is a lot of symmetry in this definition. Firstly, we have

$$\texttt{between}\ (f\ n)\ (f\ n')\ (f\ n'') \iff \texttt{between}\ (f\ n'')\ (f\ n')\ (f\ n)$$

since we can always permute the first and last arguments of the `between` predicate. Secondly, given three distinct points n, n' and n'', there are six ways to arrange them so that they are linearly ordered.

In some cases, we can take care of this latter symmetry by following a simple technique [26] in proving a suitable without-loss-of-generality theorem. Informally, our theorem says that if a formula is true regardless of how one arranges the free-variables, then we can assume a specific ordering on those variables. Formally, for the case of three variables:

$$\begin{aligned}
&(\forall m\ n.P\ m\ m\ n)\\
&\wedge (\forall m\ n\ p.P\ m\ n\ p \iff P\ n\ m\ p)\\
&\wedge (\forall m\ n\ p.P\ m\ n\ p \iff P\ n\ p\ m)\\
&\wedge (\forall m\ n\ p.m < n \wedge n < p \implies P\ m\ n\ p)\\
&\implies \forall m\ n\ p.P\ m\ n\ p.
\end{aligned}$$

The second and third conditions require that the predicate P holds up to any symmetry in its arguments. With these, the first condition requires that the predicate is true assuming that any two of its arguments are equal. Thus, in the fourth condition, we can assume that the the first argument is strictly less than the second which is strictly less than the third, and from this conclude that the predicate holds for any three natural numbers (which, we recall, are concretely represented by a sequence of geometric figures).

In practice, the idea to using this theorem is to match its conclusion to the goal, discharge the first three conditions using simplification, and then verify the remaining "without loss of generality" condition.

5.7.1 Verification

In verifying Theorem 6, we have tried to reinforce Hilbert's claim that it is a generalisation of Theorem 5 by showing how to generalise his *proof*. Up until now, we have not even considered the special case of the proof, which makes up the third and final part of the proof of Theorem 5. It is given in Figure 5.5

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.

Figure 5.5: Case-Analysis for Theorem 5

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 clearly 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 proof now, following the original proof closely to show how the inferences become generalised.

Proof. Now let $n+1$ points on a line be given. Take n of the points and label P, Q_1, $Q_2, \ldots, Q_{n-2}$, 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. In any of these cases, we can apply Theorem 5.11 to label all the points in order.

$$\texttt{bounds}\, P\, Q\, X \iff P,Q \in X \wedge \forall R. R \in X - \{P,Q\} \implies \texttt{between}\, P\, R\, Q \tag{5.10}$$

$$\begin{aligned}&\texttt{finite}\, X \wedge \texttt{ordering}\, f\, X \wedge \texttt{between}\ (f 0)\ (f(|X|-1))\ x \\ &\qquad \implies \exists g. \texttt{ordering}\, g\ (\{x\} \cup X)\end{aligned} \tag{5.11}$$

$$\begin{aligned}&x \in X \wedge \texttt{finite}\, X \wedge \texttt{ordering}\, f\, X \wedge \texttt{bounds}\ P\ Q\ X \\ &\qquad \implies \exists f'. \texttt{ordering}\, f'\ X \wedge f'\, 0 = P \wedge f'\ (|X|-1) = Q.\end{aligned} \tag{5.12}$$

□

We have replaced Hilbert's reference to his earlier parts of Theorem 5 with a reference to its generalisation in Theorem 5.11. 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 applies to the points P, Q_1, Q_2, …, Q_{n-2}, R, and the ordering g applies to the points P, Q_1, Q_2, …, Q_{n-2}, S. But there is some notational abuse here, as we gloss over an implicit assumption that $f(0) = g(0) = P$. In our verification, we can make this reasoning explicit with Theorem 5.12, which uses the auxiliary concept of `bounds`.

Our generalised proof handles its case-analyses by one application of Theorem 4, and two applications of an induction 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:

$$\exists f. \texttt{ordering}\, f\, \emptyset$$
$$\texttt{collinear}\{x,y,z\} \implies \exists f. \texttt{ordering}\, f\ \{x,y,z\}.$$

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:

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

5.7.2 Exactly Two Orderings

Hilbert closes his statement of Theorem 6 with an obvious point which we found to have a slightly challenging formal proof and 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.

$$\texttt{ordering}\,f\,X \wedge \texttt{ordering}\,g\,X \wedge f\,0 = g\,0 \wedge (\texttt{finite}\,x \implies n < |X|) \implies f\,n = g\,n. \tag{5.13}$$

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 between $f\ n$ and $f\ (n+1)$, or a point between $g\ n$ and $g\ (n+1)$. But the existence of such a point contradicts the verified fact that orderings are injections:

$$\texttt{ordering}\ f\ X \wedge (\texttt{finite}\ X \implies m < |X| \wedge n < |X|) \wedge f\ m = f\ n \implies m = n. \tag{5.14}$$

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

$$\texttt{finite}\,X \wedge \texttt{ordering}\ f\ X \wedge \texttt{ordering}\ g\ X \wedge (\texttt{finite}\ X \implies n < |X|) \implies \forall n. f\ n = g\ n \vee f\ n = g(|X| - n - 1) \tag{5.15}$$

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

5.8 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 linear arithmetic. To handle this, we will need to consider what is basically the inverse of the `ordering` function, given as f in the following theorem:

$$\begin{aligned}
&\texttt{finite}\, x \wedge \texttt{collinear}\, X \\
&\quad \Longrightarrow \exists f. \forall A\, B\, C.\, A \in X \wedge B \in X \wedge C \in X \\
&\qquad\qquad \Longrightarrow (\texttt{between}\, A\, B\, C \\
&\qquad\qquad\qquad \Longleftrightarrow (f\, A < f\, B \wedge f\, B < f\, C) \vee (f\, C < f\, B \wedge f\, B < f\, A)) \\
&\quad\ \wedge \forall A\, B.\, A \in X \wedge B \in X \Longrightarrow (A = B \Longleftrightarrow f\, A = f\, B)
\end{aligned} \tag{5.16}$$

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 equations 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 have only considered simple betweenness problems (at most, ones involving six points).

When discharging the assumption in Theorem 5.16, we expect the user to have instantiated X with a concrete set enumeration (an expression of the form $\{P_1, P_2, \ldots, P_n\}$). These sets can be proven finite by simple rewriting. To prove them collinear, we reuse our incidence discoverer from Chapter 3.

We package this up as the procedural tactic `ORDER_TAC`, which is parameterised on a concrete set enumeration. It solves its goal by instantiating X in Theorem 5.16, running an incidence discoverer as a tactic via `discover_tac` (see §3.7), and finally discharging the assumption in Theorem 5.16 by rewriting. It then hands over to HOL Light's decision procedure for linear arithmetic, `ARITH_TAC`.

5.8.1 Example: Theorem 7

To round up this chapter, we will apply our linear reasoning tactic `ORDER_TAC` to a theorem which says that there is an infinite set between any two points.

In a sense, this has been covered indirectly by Theorem 5.7, but our purpose with that theorem was to settle foundational questions about our logical assumptions, and provide us with enough expressive power to formally verify Theorem 6. Here, we can be more direct in our formalisation.

THEOREM 7. Between any two points on a line there exists an infinite number of points.

$$P \neq Q \implies \texttt{infinite}\ \{R \mid \texttt{between}\ P\ R\ Q\}. \tag{5.17}$$

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, and 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 proof, 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.

```
    ...
    so consider R such that between (f 0) R (f1)                          7
    have between P (f 0) Q ∧ between P (f 1) Q from 6 by...               8
    hence between P R Q from 6,7 using ORDER_TAC {P,Q,R,f 0,f 1}
...
```

5.9 Conclusion

In this chapter, we tackled the verification of Hilbert's Theorem 6. This is a metatheorem, so its verification requires a logic with more expressive power than is available in first-order logic.

We aim to be conservative, so while we assume the power of higher-order logic, we do not assume the axiom of infinity from which we normally acquire the natural numbers. Instead, we have shown that Hilbert's first two groups of axioms imply infinity as a theorem. On the way, we verified that the first group alone cannot do this, since it admits a finite model.

We described two ways that we can implement Theorem 6 as a tactic. We suggest this is more faithful to Hilbert's intentions, since the theorem is metatheoretical and computational, manipulating labellings according to a scheme.

Our first tactic requires settling on a way to represent stacked betweenness claims and then writing procedures to enumerate all possible stackings for a given finite list of points. Our second strategy was to use our formally verified Theorem 6 to convert betweenness problems into arithmetical problems. That way, we can reuse decision procedures for linear arithmetic.

The second tactic has the advantage that it automatically copes with problems with equations and inequalities. In our first tactic, we assume that all points are distinct when we enumerate the possible orderings, and we found that this assumption is too strong in practice. That said, we expect that our first tactic has potentially better performance, since it is more carefully tailored to Hilbert's geometry. We leave analysis of this matter to future work.

This concludes the description of the automation we have implemented for our verification: 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 were used extensively in the rest of our verification, which will be discussed in the remaining chapters.

Chapter 6

Ordering in the Plane

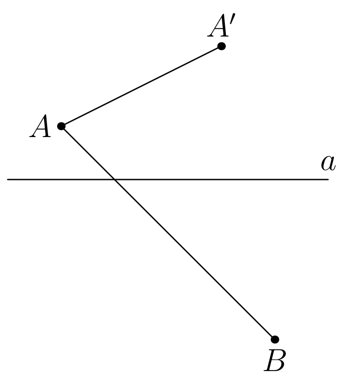
In the last chapter, we considered the theory of linear-ordering based on Hilbert's three-place `between` relation. This culminated in a tactic for solving problems relating points along a line by reducing them to a decision procedure for linear problems of natural numbers. The use of natural numbers was explicitly permitted in Veblen's ordered geometry [77], but, at least, in the last chapter, we showed how they can be recovered in higher-order logic from geometry without the axiom of infinity.

Our tactic reducing geometry problems to linear arithmetic thus allows us to reason about the relative positions of points along a line. But what if we want to reason about the relative positions of points in a *plane*? For this, we can think about a line in the plane and ask which *side* of the line a point in the plane must lie on. This idea of *sides* is introduced next in the *Foundations of Geometry*, followed by analogous ideas which culminate in the definition of a *ray*. The two ideas shall be the subject of this chapter.

6.1 Definitions and Formalisation

After stating that there are an infinite number of points on a line in Theorem 7, Hilbert gives the following theorem and definitions:

> 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.*

We are unclear why these definitions are emphasised. *Rays* appear to be an important notion for later definitions and axioms, but a potential and unweildy four-place relation `same_side` relating the point O and line a with every other pair of points did not appeal to us. Besides, once we have a notion of rays, we can just say that two points are on the line a on the same side of O when they lie on a *ray* lying on a and emanating from O.

We decided we would only formalise the notion of *ray*, and we took a similar attitude with the first definition and focused on the two-dimensional analogue of rays, namely *half-planes*. With that view, a *half-plane* is the totality of points on the same side of a line a in the plane α. Two points can then be said to lie on the same side of the line a precisely because they lie on a single half-plane in α and bounded by a.

6.1.1 Rays

As we shall see in Chapters 9 and 10, we did not find the notion of a ray a particularly useful tool in formal proofs, so in this subsection, we will need to justify keeping them in our formalisation. We believe they will be crucial 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'.*

The thing to note here is that Hilbert's axioms have increased substantially in complexity. Here, we have an axiom which juggles eight geometric entities, six of which are *derived notions*. It is easy to make a mistake in formalising this axiom if one is starting from just Hilbert's primitives. 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 keeping the definitions reasonably type-safe. We shall develop some of this theory in the current chapter, where we hope it can be used for further formalisation of Group III.

6.1.2 Quotienting

With a slight clarification, the relations which appear in Hilbert's definitions above 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*. Note that we account for all three dimensions here, and allow rays to emanate from a point in all directions. 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.

We have had to fill in an ambiguity in Hilbert's definition, and exclude from consideration the origin O for rays and the boundary line a for half-planes. If we include O and a, 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 but declare all other points to be on a different side of O and the points of 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. As an aside, 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]" [27].

6.1.3 Automatic Lifting

Now HOL Light has several powerful procedures for automatically dealing with *quotienting* and producing a strong type for the quotient sets. Assuming that $R : \alpha \to \alpha \to \alpha$ is an equivalence relation, there is a procedure which splits its domain α 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 all of these facilities to introduce the new abstract type of *rays* and *half-planes*, and introduce our primitive relations on these abstract types by lifting well-defined relations.

Unfortunately, simple type theory and Hilbert's geometry makes this highly problematic. The "same side" relations we define above are only equivalence relations on a family of subsets of space. Our equivalence relations for rays are indexed by a point O and are partial on the set of points in space minus O. Our equivalence relations for half-planes are indexed by a line a and are similarly partial on 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.

6.1.3.1 Intermediate Types

For any point O, we want to consider the set of all points P in space which are not on O. We can do this by pushing O and P into a pair and abstract their distinctness into a new type. In effect, we are defining the type of *arrows* $\overrightarrow{OP}$.

The relation "same side of" can now be reinterpreted for 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:

$$\begin{aligned} &\texttt{equiv\_arrow} : \texttt{arrow} \rightarrow \texttt{arrow} \rightarrow \texttt{bool} \\ &\texttt{equiv\_arrow}\, \overrightarrow{OP}\, \overrightarrow{OQ} \iff O = O' \wedge (P = Q \vee \texttt{between}\ O\ P\ Q \vee \texttt{between}\ O\ Q\ P) \end{aligned} \tag{6.1}$$

We now just verify that this relation is a total equivalence relation:

$$\begin{aligned} &\texttt{equiv\_arrow}\ s\ s \\ &\wedge (\texttt{equiv\_arrow}\ s\ t \iff \texttt{equiv\_arrow}\ t\ s) \\ &\wedge (\texttt{equiv\_arrow}\ s\ t \wedge \texttt{equiv\_arrow}\ t\ u \implies \texttt{equiv\_arrow}\ s\ u) \end{aligned} \tag{6.2}$$

The only potential challenge needed when proving this theorem is dealing with transitivity. In our earlier work [65], 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 (6.1) 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.

Next, we will discuss some of the development of our theory of rays. This differs from our earlier work in Isabelle since we are using quotienting procedures in HOL Light. Moreover, the theory we develop here is cleaner and has much better coverage of the important ideas. Almost all of the theorems we verified were chosen from analogous theorems in the theory of half-planes which we cover in §6.2, and in whose completeness we are confident having used the theory extensively in verifying the Polygonal Jordan Curve Theorem in Chapters 9 and 10.

6.1.3.2 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 abstraction and representation functions `mk_ray` and `dest_ray` which map between equivalence classes of arrows and the rays they represent. These functions are typical in typed theorem provers when we want to carve out a type from a subset of an existing type. They are used to prove the most elementary theorems of the new type, the ones which require reasoning in terms of the representatives.

The great thing about HOL Light's quotienting facilities is that we have no need to deal with these particular abstraction and representation functions. All formal proofs of lifted theorems will apply these functions automatically.

However, HOL Light will not automatically plumb facts 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 : (point,point)` $\rightarrow$ `arrow`.

$$\begin{aligned}&\texttt{on\_ray\_of\_arrow}\ P\ \texttt{a}\\ &\quad\Longleftrightarrow P \neq \texttt{arrow\_origin}\ a \\ &\qquad\quad \wedge\, \texttt{equiv\_arrow}\ a\ (\texttt{mk\_arrow}\ (\texttt{arrow\_origin}\ a, P))\end{aligned} \tag{6.3}$$

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 the fact that

$$\texttt{on\_ray\_of\_arrow}\ P\ \overrightarrow{OQ} \Longleftrightarrow P = Q \vee \texttt{between}\ O\ P\ Q \vee \texttt{between}\ O\ Q\ P \tag{6.4}$$

This could potentially be fixed by extending HOL Light's quotienting facilities to handle indexed families of equivalence relations, and we would like to propose such extensions as further work. As it stands, we believe that Hilbert's geometry offers a nice example of where such facilities would be useful.

For completeness, we briefly mention the intermediate type for half-planes and the equivalence relation we define on this intermediate type. We will 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.

Our 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.

$$\begin{aligned}
&\texttt{equiv\_half\_plane}\ (P,a)\ (Q,b) \\
&\quad\iff a = b \\
&\qquad\quad \wedge (\exists\alpha.\texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha \\
&\qquad\qquad \wedge \forall P.\texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha) \\
&\qquad\quad \neg\texttt{on\_line}\ P\ a \wedge \neg\texttt{on\_line}\ Q\ a \\
&\qquad\quad \neg(\exists R.\texttt{on\_line}\ R\ a \wedge \texttt{between}\ P\ R\ Q)
\end{aligned} \tag{6.5}$$

Note that the validity of this definition will be determined, not by its opaque definition, but by the list of theorems which specify it, and by the fact that half-planes proved to be a crucial scaffold in our verification of the Polygonal Jordan Curve Theorem. We consider this theory of half-planes next.

6.2 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 another story. We get this impression from Hilbert himself, who justifies the definition of half-planes with a distinguished theorem (Theorem 8), but merely *assumes* his definition of rays is sound.

As with the theory of rays, our theory is based on lifting from an intermediate type. All of the resulting theorems are providing in Appendix B. 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`. We trust that the theory developed is complete, in as much as all theorems about half-planes can be derived as if `half_plane` were entirely abstract. We have some evidence for this given that we have used the theory to verify a major theorem of ordered geometry, the Polygonal Jordan Curve Theorem without ever having to unfold our `half_plane`*s* (see Chapters 9 and 10).

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, this can be understood as a strengthening of Pasch's Axiom (II, 4).

6.2.1 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 (6.5), 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 (6.5). 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 6.1.

Now we had proven this theorem in our earlier work in Isabelle [65], but there, we derived two lemmas whose proofs are based on numerous messy case-splits. In our new formalisation, these messy details vanish. We take this as evidence, along with our short verification of transitivity for rays in §6.1.3.1, that our linear reasoning tactic is relieving us of some burden.

In the proof in Figure 6.1, 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 then manually adding in facts until a contradiction is found, 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.

6.2.2 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 (6.5), 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 was the case in the first-edition, where Hilbert explicitly uses "either...or" (and similarly in the German edition). By the tenth edition, the "either" has disappeared, and Hilbert now intends the inclusive-or, since he remarks that the case of a intersecting both sides can be refuted. Nevertheless, 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.

```
assume on_line $P\ a \wedge \neg$on_line $C\ a \wedge$ between $A\ P\ B$                0
assume $\exists a.$on_line $A\ a \wedge$ on_line $B\ a \wedge$ on_line $C\ a$           1
take $P$
thus on_line $P\ a$ from 0
have $C \neq P$ from from 0
hence between $A\ P\ C \vee$ between $B\ P\ C$ using ORDER_TAC $\{A,B,C,P\}$ from 0, 1
```

Figure 6.1: Pasch's Axiom when A, B and C are collinear

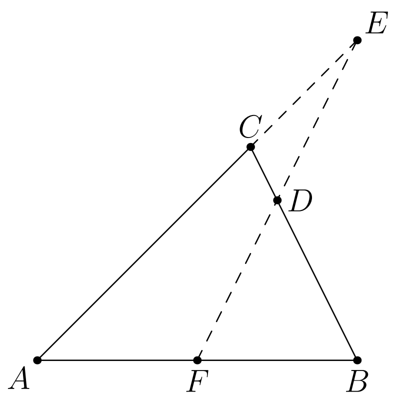


Figure 6.2: Supplement I

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 Pasch's Axiom to conclude that C must lie between A and E, contradicting our assumptions. The diagram shows the situation in Figure 6.2, and shows how this step corresponds precisely to a use of the *outer Pasch* version of the axiom (4.3).

The formal proof is shown in Figure 6.3. Our incidence discoverer from Chapter 3 helps keep the proof steps very close to 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 Pasch axiom (4.3), 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 form a triangle, 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:

$$\begin{aligned}&\neg\texttt{on\_line}\ A\ a \wedge \neg\texttt{on\_line}\ B\ a \wedge \neg\texttt{on\_line}\ C\ a\\ &\wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a\\ &\implies \neg\texttt{between}\ A\ D\ B \vee \neg\texttt{between}\ A\ E\ C \vee \neg\texttt{between}\ B\ F\ C\end{aligned} \tag{6.6}$$

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.

6.3 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

```
assume $\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)$    0
        $\texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a$    1
assume $\texttt{between}\ A\ D\ B \wedge \texttt{between}\ A\ E\ C \wedge \texttt{between}\ B\ F\ C$    2
obviously have $D \neq E \wedge D \neq F \wedge E \neq F$ from $0,2$
hence $\texttt{between}\ D\ E\ F \vee \texttt{between}\ D\ F\ E \vee \texttt{between}\ F\ D\ E$ from $1$ by (Theorem 4)    3
have $\forall A'\ B'\ C'\ D'\ E'\ F'.\neg(\exists a.\texttt{on\_line}\ A'\ a \wedge \texttt{on\_line}\ B'\ a \wedge \texttt{on\_line}\ C'\ a)$
    $\texttt{between}\ A'\ F'\ B' \wedge \texttt{between}\ A'\ E'\ C' \wedge \texttt{between}\ B'\ D'\ C'$
     $\implies \neg\texttt{between}\ E'\ D'\ F'$ proof
    fix $A', B', C', D', E', F'$
    assume $\neg(\exists a.\texttt{on\_line}\ A'\ a \wedge \texttt{on\_line}\ B'\ a \wedge \texttt{on\_line}\ C'\ a)$    4
    assume $\texttt{between}\ A'\ F'\ B' \wedge \texttt{between}\ A'\ E'\ C'$
        $\wedge \texttt{between}\ B'\ D'\ C' \wedge \texttt{between}\ E'\ D'\ F'$    5
    obviously consider $G$ such thatbetween $A'\ G\ E' \wedge \texttt{between}\ B'\ D'\ G$
        by $(4.3), (\text{II}, 1)$ from $4,5$
    obviously qed from $4,5$ by $(\text{II}, 3)$
qed from $0,2,3$ by $(\text{II}, 1)$
```

Figure 6.3: Proof for Supplement I

a plane α and a line a in α. We find a point B on a, and a planar point A off the line a. Then, with Axiom II, 2, we can extend the segment AB through the line a to find a point C on the other side. These two points will lie in distinct half-planes. Note that this proof requires that the point B always exists, which is a consequence of Theorem 2.2.4 from Chapter 2. 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 a number of additional lifted functions, such as `line_of_half_plane` and the incidence relation `on_half_plane`, whose specifications are given in Appendix B.

$$\begin{aligned}
&(\forall P. \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha)\\
&\quad\implies \exists hp\ hq.\ hp \neq hq\\
&\qquad \wedge a = \texttt{line\_of\_half\_plane}\ hp \wedge a = \texttt{line\_of\_half\_plane}\ hq\\
&\qquad \wedge (\forall P. \texttt{on\_plane}\ P\ \alpha\\
&\qquad\qquad \iff \texttt{on\_line}\ P\ a \vee \texttt{on\_half\_plane}\ hp\ P \vee \texttt{on\_half\_plane}\ hq\ P)
\end{aligned} \tag{6.7}$$

Note that we have had to break convention with our order of arguments for the incidence relation. While relations such as `on_line` have type `point` $\rightarrow$ `line` $\rightarrow$ `bool`, our half-plane incidence relation has the flipped type `half_plane` $\rightarrow$ `point` $\rightarrow$ `bool`. We can understand why we have to put the `point` type in second position when we consider exactly what we are saying is well-defined. It is not the two-place relation `on_half_plane`, but the *set* of points incident with a half-plane. This set is well-defined regardless of our choice of representative in the intermediate type.

Now as predicates, point-sets are functions of type `point` $\rightarrow$ `bool`, and it is a function of this form which we must verify as well-defined. We obtain the function by partial application of our intermediate incidence relation:

$$\begin{aligned}
&\texttt{same\_half\_plane}\ x\ y\\
&\quad\implies \texttt{on\_half\_plane\_intermediate}\ x = \texttt{on\_half\_plane\_intermediate}\ y
\end{aligned} \tag{6.8}$$

Consequently, the type `point` must appear last in the type of our half-plane incidence relation. This will explain our potentially confusing choice of formulation in the next few chapters.

6.4 Conclusion

In this chapter, we have given an overview of some of the challenges involved in developing a theory of rays and half-planes using HOL Light's quotienting facilities. We would like to explore in more detail how we could automate the quotienting of partial relations as we have done in this chapter, and what technical difficulties would be involved in doing this. For now, our solution has been to create an abstract intermediate type so that the equivalence relations become total.

For half-planes, we noted that the principle justification for Hilbert's Theorem 8 can be understood purely in terms of strengthening Pasch's Axiom (II, 4). We just drop one of its hypotheses and render its disjunctive conclusion as an exclusive-or. In earlier editions, this exclusive-or was originally part of the axiom, but when it was dropped, Bernays felt that a proof of how it would be recovered needed spelling out. We would argue the same for Theorem 2.2.4. This is also needed to prove Theorem 8, and it was originally part of the axioms in the first edition. Now that it has been factored out, it would be useful to have its proof as another supplement to the text.

Much of the issues discussed so far in Group II go some way to reinforcing the idea that Hilbert was not particularly concerned with ordered geometry. His axioms were highly redundant, and theorems such as Theorem 8 are left without proof even as they require a fair amount of supplementary details.

Hilbert is far more preoccupied with developing a theory of segment and angle congruence in Group III, which makes up the bulk of his axiomatics. To help bring the point home, consider that Hilbert is happy to spell out the entirely trivial when it comes to working with congruence. He devotes three paragraphs to explaining that segment congruence is an equivalence relation, stressing that "it is by no means obvious that **every segment is congruent to itself.**" This is quite a biased presentation. Hilbert omits all the the *far* trickier detail needed to prove that his "same-side" relation for half-planes is an equivalence relation, and that it correctly partitions the plane.

The presentation gets substantially weaker for Hilbert's next theorem, which he gives without any proof and yet which requires the most difficult we have considered so far. The Polygonal Jordan Curve Theorem is the subject of the next four chapters.

Chapter 7

The Jordan Curve Theorem for Polygons

In this chapter, we shall discuss the next theorem we encounter in Hilbert's *Foundations of Geometry*, which appears as Theorem 9 in the 10th edition. This theorem is a special case of the Jordan Curve Theorem, which applies only to simple polygons. Our discussion will draw together a number of historical threads and characters connected to the *Foundations of Geometry*, and point out interesting subtleties and obstacles that come with trying to rigorously prove the theorem from Hilbert's very weak axiom system. We shall consider several informal proofs, including Veblen's 1904 proof, which we believe to contain a major fault. The verification of the theorem is left to Chapters 9 and 10.

In sections 7.1 and 7.2, we give a short historical introduction to the polygonal case in the context of the more general theorem. In §7.3, we give Hilbert's formulation of the theorem and, because Hilbert did not supply a proof, we provide a detailed one of our own in §7.4. Finally, in §7.5, we present another proof given by Veblen [77]. This latter proof is probably incorrect, but it *is* based on axioms very close to Hilbert's own, and contains useful ideas. Our final verification in Chapters 9 and 10 is based on our diagnosis of the problems with Veblen's proof.

7.1 Relationship with the Full Jordan Curve Theorem

The full Jordan Curve Theorem 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 [33]:

> 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.

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* [34], and to this day, the details are regarded as quite involved. 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*, and 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 the first correct proof was provided by Veblen in 1905 [78]. Both Veblen and folklore point out that Jordan had to assume the polygonal case, which should have been proven as a lemma. Hales, on the other hand, has formally verified the theorem in HOL Light, and put together a strong defence of Jordan [20], 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* [17], he repeats the folklore that Veblen gave the first correct proof, and claims that even the polygonal case is “devilishly difficult to prove.”

We might suggest that this controversy begins with Hilbert’s 1899 edition of *Foundations of Geometry*. 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 [18]. 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 §6.2.2). That would have taken more than a few supplementary remarks.

Now Veblen himself, 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 [77], and in this 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[1], was contributing proofs to later editions of Hilbert's text (see §4.3). 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*. It seems plausible to us, therefore, that Veblen's criticism of Jordan is explainable by his particular standards of rigour and the context of an axiomatic theory of geometry. When it came to his theory of order, his standards of rigour were even higher than Hilbert's.

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 [20], the generality was refuted ten years later. R. L. Moore, who had been a student of Veblen's. R. L. Moore showed that, according to Veblen's axioms, all of his planes are homeomorphic to the Euclidean plane [55].

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 §7.5, a correct proof seems to have eluded Veblen himself.

[1]Not to be confused with Veblen's student R. L. Moore.

7.2 Generality of the Polygonal Case

We can discuss the problems raised by our level of generality by considering the two proofs of the polygonal case from the book *What Is Mathematics?* [62]. Hales mentioned these to highlight the triviality of the polygonal case. The first of these 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 specified direction. 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 probably a non-starter. At Group II, we have no theorems which say that parallel lines even *exist*, nor do we have a theorem which says we can cast rays in a given direction. The needed theorems are actually developed in Group III and Group IV, where angle congruence is introduced as a primitive, and where we can start manipulating *directions*. 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 [75], 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 naive formulation of this argument in terms of angles and continuous motion will, for the same reasons as above, be well outside the scope of Hilbert's first two groups of axioms.

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. But worse, 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 par-

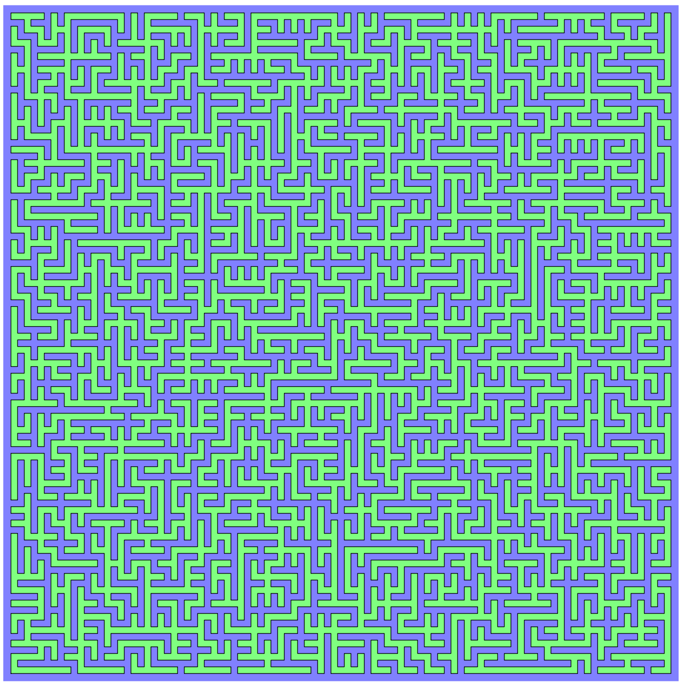


Figure 7.1: A Simple Polygon

allel 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.

7.3 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 maximal 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, and thus the theorem. Having already defined a segment as an unordered pair of points, Hilbert can now define a *polygonal segment* as follows[2]:

> 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

[2]Veblen calls these *broken lines*.

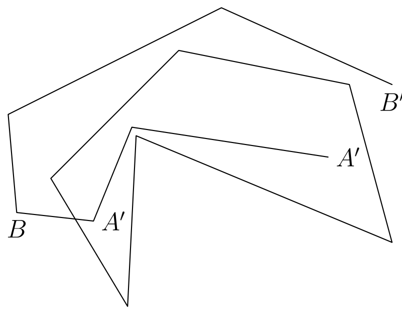
> will also be briefly denoted by $ABCD \ldots KL$. The points inside the segments AB, BC, CD, ..., KL as well as the points A, B, C, D, ..., K, L are collectively called the *points of the polygonal segment*.

Polygons are then polygonal segments where the points A and L coincide. We then 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*.

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 labeling 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.



In the remainder of this chapter, we shall identify a polygon with its list of vertices $P_1P_2P_3 \cdots P_n$.

First, we shall briefly summarise the last few sections: as with Hilbert's *Foundations of Geometry*, the formulation and proof of the Jordan Curve Theorem marks a significant development in the history of rigorous, axiomatic mathematics and geometry. Its position in Hilbert's text means that it should be possible to formulate and prove it in very basic terms and from very simple axioms, based on ideas about half-planes. It is somewhat controversial whether this is trivial, but we hope that the next few chapters will go some way towards clarifying the matter.

7.4 Point-in-Polygon Proof

One feature of Veblen's proof, and the proof we verify in Chapter 9, is that the treatment of the two regions defined by a simple polygon is symmetric. This means, however, that unlike the plumb-line and winding number proofs from *What is Mathematics?*, Veblen's proof does not hint at any sort of point-in-polygon *test*, one that could be implemented on a machine.

Before we encountered Veblen's proof, we had developed one of our own which does allow for such a test, which reduces the problem of deciding whether a point is inside a polygon to the problem of deciding whether a point is on a given side of the polygon's diagonals. The proof is inductive over the vertices of a simple polygon, and goes some way towards thinking of the Jordan Curve Theorem for polygons as a triangulation problem.

We decided to try an inductive proof on the total number of vertices of a simple polygon. To do this, it is necessary to show that every simple polygon can be broken down into smaller, simple polygons, until one reaches the smallest possible polygon (the triangle). We have verified the Polygonal Jordan Curve Theorem in this base case as part of our main verification in Chapters 9 and 10.

The idea that we might be triangulating polygons in this proof should give the careful reader pause, since most proofs that a polygon can be triangulated assume the Jordan Curve Theorem for polygons [49]. We are thus in danger of creating a circular argument. We avoid this danger because firstly, at each stage, we do not require that the smaller polygons are subsets of the original (thus, we do not actually identify a triangulation). And secondly, we are always able to use the Jordan Curve Theorem as an inductive hypothesis at each stage.

To split a given simple n-gon where $n > 3$ into two simple polygons, we just pick a line connecting two non-adjacent vertices, a *diagonal*, which does not intersect the polygon. The simplest example of interest is the concave quadrilateral shown in Figure 7.2. This quadrilateral has exactly two diagonals. The diagonal P_2P_4 lies in the interior of

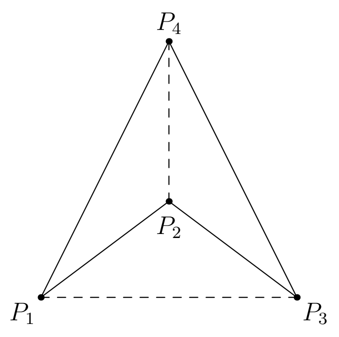


Figure 7.2: Concave Quadrilateral

the polygon, while the diagonal P_1P_3 lies in the exterior.

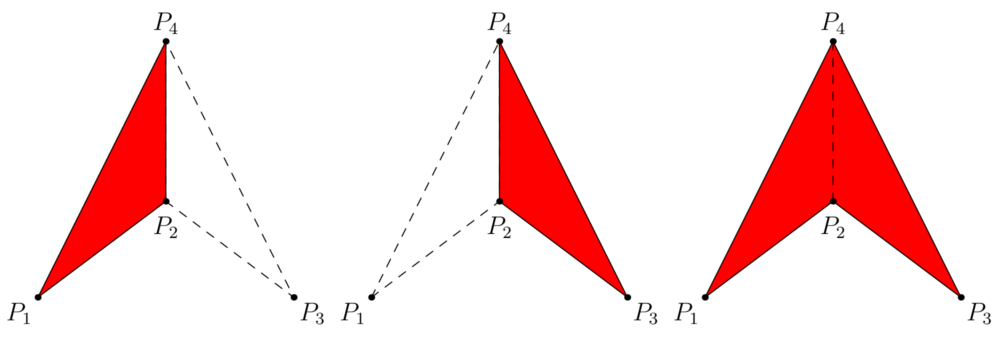
If we take the diagonal P_2P_4, we see that the interior of the quadrilateral consists of the union of the interiors of two triangles, namely $P_1P_2P_4$ and $P_2P_3P_4$, together with the diagonal P_2P_4 itself (see Figure 7.3(a)). If we take the diagonal P_2P_4, then we see the interior of the quadrilateral consists of the interior of the triangle $P_1P_3P_4$, minus the interior of $P_1P_2P_3$ and the boundary of $P_1P_2P_3$. Thus, we can define the interior of the quadrilateral in terms of the interiors of two triangles. We shall extend this to the general case for an n-gon with $n > 3$. First, we show how to find diagonals.

7.4.1 Finding a Diagonal

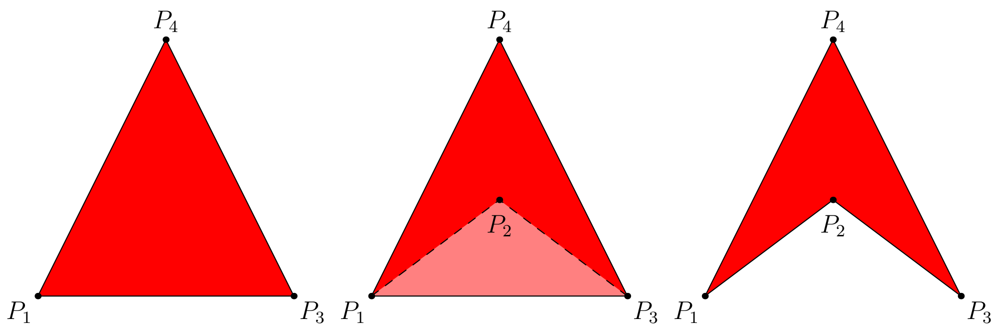
This subsection contains all the core ideas of the proof. This proof has not been verified and we include it for aesthetic reasons only. We are confident that it could be verified without much difficulty, reusing much of the verification discussed in chapters 9 and 10. We would like to verify this proof as further work, since a triangulation proof allowing a point-in-polygon test appeals to us. The section is quite dense, and can be skipped.

A simple polygon $P_1P_2 \ldots P_n$ where $n > 3$ has at least one diagonal which does not intersect the polygon. To see this, we assume without loss of generality that $P_1P_2P_3$ are non-collinear, and that P_n is not inside the triangle $P_1P_2P_3$. Consider the line P_1P_3. If this line does not intersect the polygon, then it is a suitable diagonal. Otherwise, take the vertex P_m where $3 < m < n$ such that the line P_1P_m intersects P_2P_3 in a point X and such that, for any other $P_{m'}$ where $3 < m' < n$, if $P_1P_{m'}$ intersects P_1P_2 at X', then X is between P_1 and X'. We then have that P_1P_m is the required diagonal (see Figure 7.4), which yields two smaller simple polygons, shown in green and blue.

A *very* slightly modified version of this argument is a core component in the proof we



(a) Union of Two Triangles



(b) Difference of Two Triangles

Figure 7.3: Decomposing a Concave Quadrilateral

actually verified. We shall explain it in more detail in §10.3.2.

7.4.2 Cases

As noted, a diagonal D which does not intersect its polygon p divides that polygon into two smaller polygons p_1 and p_2. Generalising the situation from Figure 7.3, we have that if D is interior to p, then the interior of D can be defined as the union of the interiors of p_1 and p_2, together with D. On the other hand, if D is exterior to p, and the interior of p_2 is a subset of the interior of p_1, then we can define the interior of p as the interior of p_1 minus the boundary and interior of p_2. It therefore follows that the interior of a polygon is formed by recursively adding and subtracting out the interiors of smaller polygons. Finally, at each step, we define the exterior of the polygon as the plane minus the interior and minus the polygon's boundary.

There is obviously circularity here, since whether or not D is an interior or exterior diagonal should be a *corollary* of our argument. To remedy this, we shall need to characterise the two cases some other way.

In Figure 7.5, we illustrate the two cases by showing a fragment of a polygon and

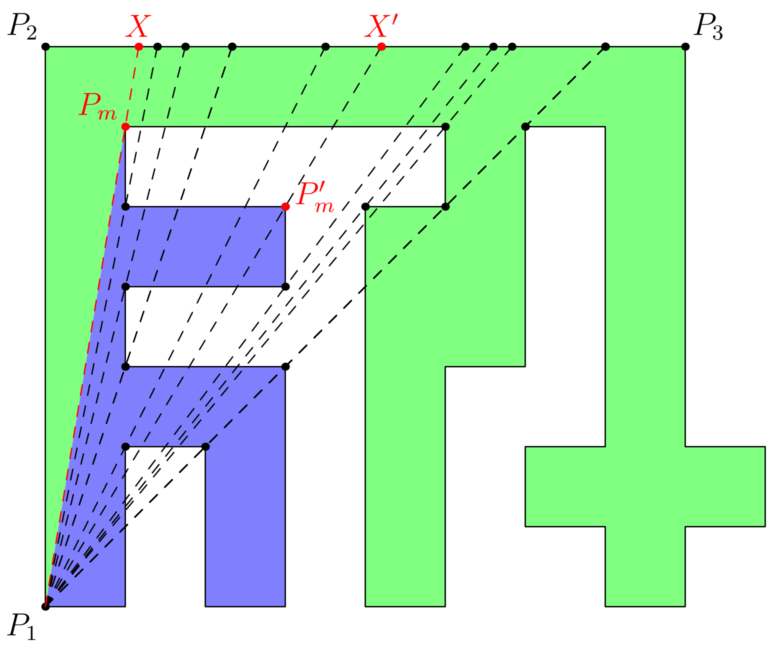


Figure 7.4: Finding a Diagonal

its diagonal. Here, we assume that the polygon p is of the form P, P_1, P_2, P_3, P_4, P_5, Q, Q_1, Q_2, Q_3, Q_4, Q_5 and has a diagonal PQ which does not intersect p. Our recursive step involves splitting the polygon into $p_1 = P, P_1, P_2, P_3, P_4, P_5, Q$ and $p_2 = Q, Q_1, Q_2, Q_3, Q_4, Q_5, P$.

We now take a point X on the diagonal PQ, and we cast rays out to the points Y and Z on either side of PQ, such that X is the only point of intersection of the segment YZ and the polygons p_1 and p_2 (ray-casting will be discussed in §10.3.1).

We now make the following inductive hypotheses:

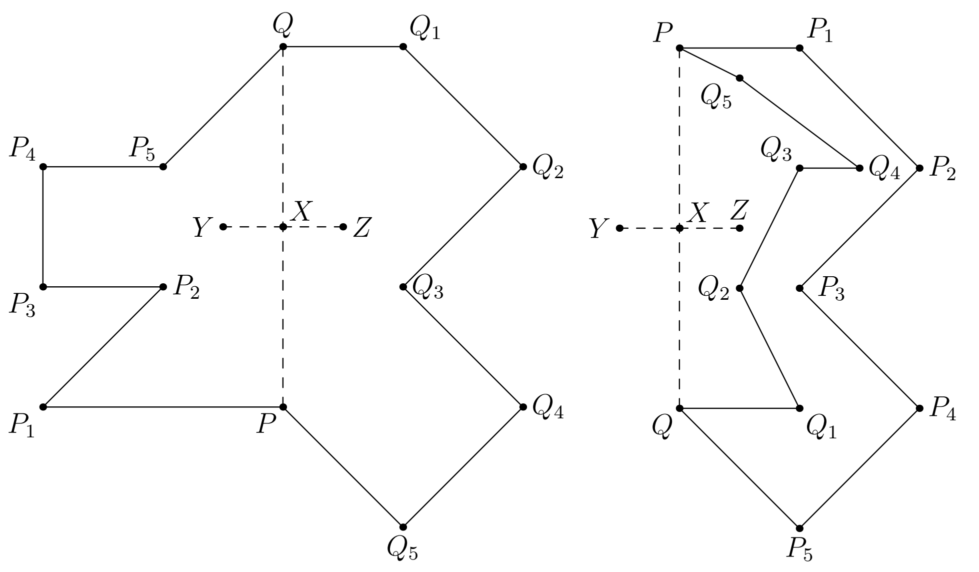
IH1 Interior points A and B of p_1 can be connected by a polygonal path which does not intersect p_1, and similarly for p_2.

IH2 Exterior points A and B of p_1 can be connected by a polygonal path which does not intersect p_1, and similarly for p_2.

IH3 Any path connecting an interior point A of p_1 to an exterior point B of p_1, must intersect p_1, and similarly for p_2.

IH4 If Y' and Z' are endpoints of a segment which crosses exactly one side of p_1, then they lie in different regions with respect to p_1 (and similarly for p_2).

IH5 If another segment $Y'Z'$ does not intersect any side of p_1, nor any side of p_2, then Y' and Z' lie in the same region.



(a) Union for Interior Diagonal

(b) Subtraction for Exterior Diagonal

Figure 7.5: Diagonal Cases

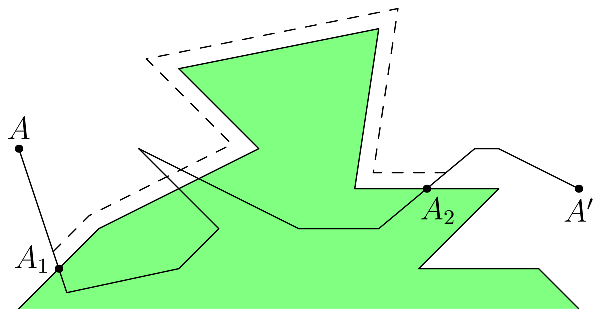


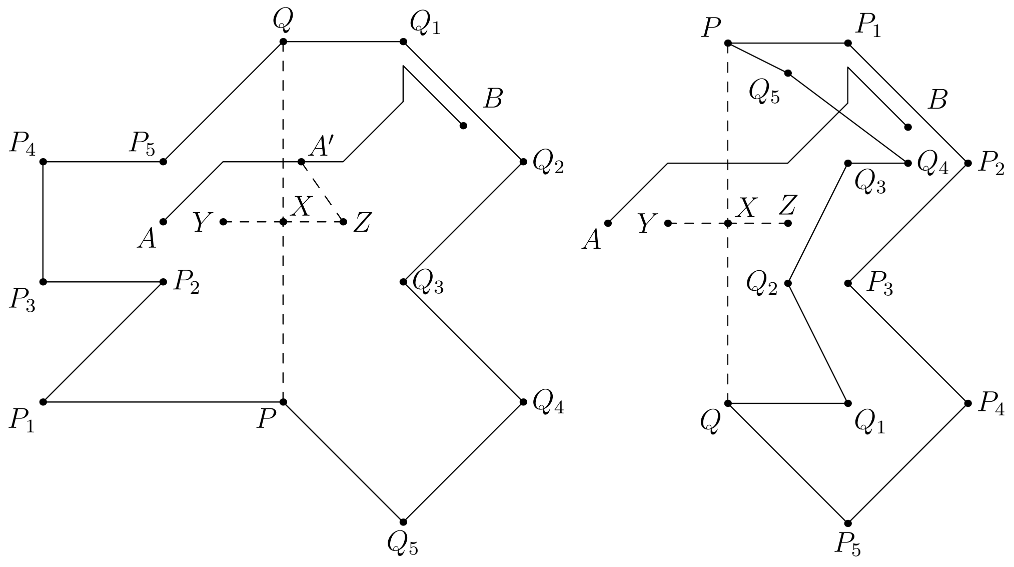
Figure 7.6: Navigating around the Exterior

Assuming that interiors and exteriors are non-empty and cover the plane, the first three hypotheses are equivalent to the polygonal Jordan Curve Theorem. They give us the following two cases:

1. Both Y and Z lie in the interior of exactly one of the polygons p_1 and p_2 (see Figure 7.5(a)).

2. One of Y and Z lies in the exterior of both p_1 and p_2 (see Figure 7.5(b)).

The first case corresponds to PQ being an interior diagonal. The second case corresponds to PQ being an exterior diagonal.



(a) Union Case (b) Subtraction Case

Figure 7.7: Paths for IH3

7.4.3 Union Case

For the case depicted in Figure 7.5(a), we will assume, without loss of generality, that Y lies in the interior of p_1 and Z lies in the interior of p_2. We now define the interior of p as the union of the interiors of p_1 and p_2 and the interior of the diagonal PQ. We define the exterior as the set of points not on the polygon and not in the interior. We just need to prove IH1–IH5 for the polygon p.

IH1 Interior points A and B of p can be connected by a polygonal path which does not intersect p.

Proof. Suppose A and B both lie inside p_1, or both lie inside p_2. We apply IH1 to p_1 and p_2, and thereby connect A and B with a polygonal segment lying in the interior of p. On the other hand, if we suppose that A is inside p_1 and B is inside p_2, we apply IH1 to p_1 in order to connect A to Y, and we apply IH1 to p_2 in order to connect B to Z. The required path can then be completed with the segment YZ. □

IH2 Exterior points A and B of p can be connected by a polygonal path which does not intersect p.

Proof. We must have that A and B lie in the exteriors of both p_1 and p_2. By IH2 applied to p_1, there is a path through the exterior of p_1 which connects A and B. If this path lies entirely in the exterior of p_2 then it also lies entirely in the exterior of p and we are done.

Otherwise, we take the first point A_1 and the last point A_2 at which the path crosses the boundary of p_2. We then find a new path which follows the edges of the polygon until it connects these two points. See Figure 7.6 for an illustration and see §7.4.5 for some discussion on this part of the proof. □

IH3 Any path connecting an interior point A of p to an exterior point B of p, must intersect p.

Proof. Assume that A lies in the interior of p_1 and B lies in the exterior of both p_1 and p_2, and suppose there is a path connecting A and B which does not intersect p. By IH3 applied to p_1, this path must intersect the diagonal PQ. Take the last point at which it intersects this diagonal. Then there is a later point A' on the path which we can connect to either Y or Z without crossing PQ. By IH5 applied to p_1 and p_2, it then follows that A' is interior to either p_1 or p_2, whilst being connected to B. Hence, by IH3 applied to p_1 and p_2, the path must intersect p. See Figure 7.7(a) for an illustration and see §7.4.5 for further discussion. □

IH4 If Y' and Z' are endpoints of a segment which crosses exactly one side of p, then one point is interior while the other is exterior with respect to p.

Proof. Suppose $Y'Z'$ crosses a side of p. Then if we apply IH4 to p_1, we have that Y' and Z' lie in opposite regions with respect to p_1. Moreover, one of the points, say Y', lies inside p. Aiming for a contradiction, we suppose that Z' also lies inside p. Since it does not lie inside p_1, nor does it lie on PQ, it follows that it must lie inside p_2. But then if we apply IH5 to p_2, we find that $Y'Z'$ must cross a side of p_2. Hence, $Y'Z'$ crosses a side shared by both p_1 and p_2. This contradicts the fact that p is simple and PQ does not intersect either p_1 or p_2. □

IH5 If a segment $Y'Z'$ does not intersect p, then Y' and Z' lie in the same region.

Proof. If the segment does not intersect any side of p and does not intersect any sides of p_1 or p_2, then the case reduces to IH5 applied to p_1 and p_2. If it does intersect p_1 or p_2, then the side intersected must be the diagonal PQ. In this case, we can find a path which does not intersect p_1 and which connects Y to Y', or a path which does not intersect p_2 and which connects Y to Z'. Thus, Y' is interior to one of p_1 and p_2 by IH5 and thus interior to p. The same argument applies to Z'. See §7.4.5 for some further discussion. □

7.4.4 Subtraction Case

The second case differs from the first in that we lose the symmetry between p_1 and p_2. One of these polygon's interiors must be contained entirely by the other's.

We can determine which by considering whether P_1 lies inside p_2 or whether Q_1 lies inside p_1. To see why these are alternatives, suppose that P_1 does not lie on p_2 (as depicted). Then it follows from IH5 and the fact that p is simple that the boundary of $p_2 - PQ$ lies outside p_1. By considering a path interior to p_2 which connects Z and Q_1, we can therefore conclude, again by IH5, that Q_1 must be interior to p_1. Again, see §7.4.5 for further details.

We now assume, without loss of generality, that Q_1 is interior to p_1 as depicted. We then define the interior of p to be the interior of p_1 minus the interior of p_2 and its boundary.

Note that by IH5 and the fact that p is simple, we have that all points on $p_2 - PQ$ must lie inside p_1, and that all points on $p_1 - PQ$ must lie outside p_2. We now prove IH1–IH5 for the step case.

IH1 Interior points A and B of p can be connected by a polygonal path which does not intersect p.

Proof. An interior point of p is an interior point of p_1 which is not interior to p_2. If we apply IH1 to p_1, we can find a path between A and B through the interior of p_1. It is possible that this path intersects p_2, and so we must appeal to the same argument used for IH2 in the previous section and depicted in Figure 7.6, in order to navigate through the exterior of p_2. □

IH2 Exterior points A and B of p can be connected by a polygonal path which does not intersect p.

Proof. This argument is similar to that given for IH1 in the previous section. If both A and B lie in the exterior of p_1, then we know from IH1 applied to p_1 that they can be connected by a path exterior to p_1. And since $p_2 - PQ$ is interior to p_1, we know that this path does not intersect p.

Similarly, if A and B lie in the interior of p_2, we know from IH1 that they can be connected by a path interior to p_2. And since p_1 is not interior to p_2, we know that this path does not intersect p.

Finally, if A is exterior to p_1 and B interior to p_2, then we can join A to Y by a path exterior to p_1 and p_2, and join B to Z by a path interior to p_1 and p_2. The segment YZ then completes a path connecting A to B which does not intersect p. □

IH3 Any path connecting an interior point A of p to an exterior point B of p, must intersect p.

Proof. The proof is similar to that for the previous section. By definition, A lies inside p_1 and outside p_2. The point B, on the other hand, might lie inside p_2, outside p_1, or it might lie on the diagonal PQ. In any case, we can conclude that the path intersects the diagonal PQ, by applying IH3 to one of p_1 and p_2. We now take the first point at which the path intersects the diagonal. Then there will be an earlier point A' on the path which must either lie outside p_1 or lie inside p_2. This point is connected to A by a path which does not intersect PQ, and thus, by applying IH3 to one of p_1 and p_2, we can find a point where the path intersects p. See Figure 7.7(b) for an illustration. □

IH4 If Y' and Z' are endpoints of a segment which crosses exactly one side of p, then they lie in different regions with respect to p.

Proof. Suppose that $Y'Z'$ intersects a side of p_1. If we apply IH4 to p_1, we find that one of these points must be interior to p_1 and the other exterior. Moreover, since the point of intersection of $Y'Z'$ with p_1 is exterior to p_2, we can apply IH5 to p_2 and conclude that Y' and Z' are both exterior to p_2. It follows by the definition of p that Y' and Z' are then in opposite regions. A similar argument applies if $Y'Z'$ intersects a side of p_2. □

IH5 If a segment $Y'Z'$ does not intersect p, then Y' and Z' lie in the same region.

Proof. If $Y'Z'$ does not intersect p_1 or p_2, then this reduces to the case IH5 applied to these polygons. Otherwise, we suppose that $Y'Z'$ intersects the diagonal PQ.

If Y' is inside p, then it lies inside p_1 and outside p_2, and thus Z' must be outside p_1 by IH4 applied to p_1. But this means that both Y' and Z' lie outside p_2, which contradicts IH5 applied to p_2.

On the other hand, if Y' is not inside p, then there are two possibilities for its location:

The point Y' lies inside p_1 and p_2 In this case, we can apply IH4 to p_2 and conclude that Z' is outside p_2. By definition, both points are then exterior to p.

The point Y' lies on PQ In this case, Z' must also lie on PQ. Again by definition, both points are then exterior to p.

The point Y' lies outside both p_1 and p_2 Here, we apply IH4 to the point p_2, and thus conclude that the point Z' is inside p_2. By definition, both points are then exterior to p.

□

7.4.5 Discussion

There are a few definite gaps in this first proof attempt. As said, we neglected to prove the base case for triangles, and we have not shown that the recursively defined interiors and exteriors are always non-empty. Finally, the without-loss-of-generality assumption made in finding the diagonal of a polygon needs to be carefully justified (it amounts to showing that a polygon cannot have a spiral shape). We consider these to be fairly straightforward matters, however.

This is not the proof we chose to verify, though the one we *have* verified shares many of its inferences. When we come to verify those inferences, we shall see that many details above have been glossed over. We hope, then, to convince the reader to be quite wary of the above proof as it stands. The details that we have glossed over will be put on a more solid footing when we discuss our formal verification.

For instance, in Figure 7.6, we claim to be able to navigate around the exterior of a simple polygon. This intuitive idea will be needed again in our verified proof, where it needs to be carefully formulated, and the formulation needs to be carefully tied back to the premises and the desired conclusion. The details are given in §10.4.

At several places in the above proof, we also assumed we could always find a path connecting the points Y and Z in Figures 7.5(a) and 7.5(b) to various other points. A careful formulation of what characterises such points, together with a proof that the required path can be exhibited, is given in §10.7.2.

We have retained the above proof because of its computational properties. It recursively *builds* the interior of a polygon from the interiors of triangles, and thus describes the algorithm for a point-in-polygon test by reducing the problem to point-in-triangle tests, which, in turn, reduce to point in half-plane tests. Our *verified* proof does not have this feature. It was inspired by Veblen's proof, and its basic structure and focus are very different. Nevertheless, it is interesting that a number of the key steps overlap, and we shall refer back to this proof when we discuss the verification of the polygonal Jordan Curve Theorem in chapters 9 and 10.

7.5 Veblen's Proof

In his 1904 doctoral thesis [77], 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. Unlike Hilbert, he provided a complete proof for the polygonal Jordan Curve Theorem, and we shall discuss that proof in this section, comparing it to the one we sketched in §7.4.

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 theorem 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 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 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 questionable whether the proof is even correct. According to Reeken and

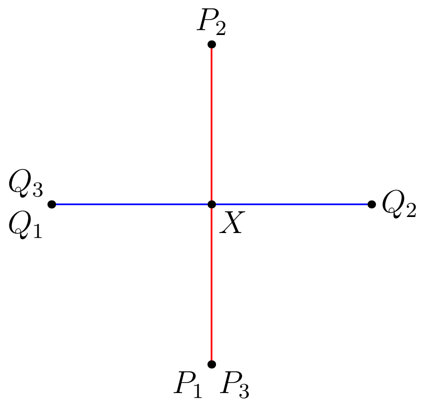


Figure 7.8: Degenerate polygons intersecting at a multiple point

Kanovei [36], 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 [19]. We could not find this criticism in Lennes [42], and were initially satisfied by Veblen's proof, and so we attempted a verification of assertion (1) above. Eventually, as one might expect, we hit an obstacle. In the next few sections, we will try to explain the difficulty.

7.5.1 Veblen's Lemma 1

The first half of Veblen's proof is given as the corollary to a very general theorem about polygons, expressed in terms of "multiple points". These are points, should they exist, where a polygonal self-intersects:

> Lemma 1. *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.*

The lemma can be used to obtain the first half of the polygonal Jordan Curve Theorem, as we explain in §9.7, but this is already a very general theorem, and one we draw special attention to in Chapter 9. We can see that it holds even for very degenerate versions of polygons. Consider the example in Figure 7.8. 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 lemma, 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.

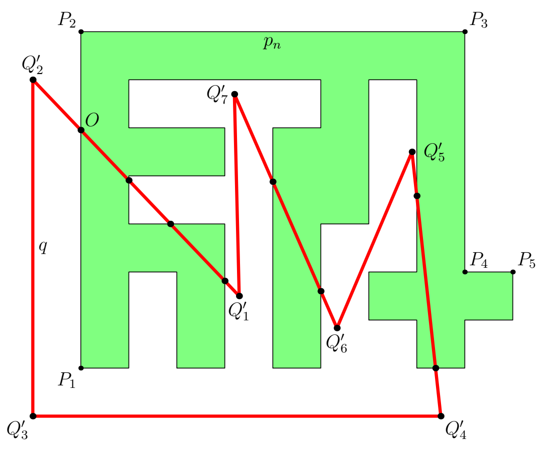


Figure 7.9: Intersections on a simple maze

We have formally verified that this particular lemma follows from Hilbert's axioms (see §9.6.2), but we gave up trying to reproduce Veblen's argument, and believe it to be invalid. 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 7.9. 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 $Q'_1Q'_2$ where the segment Q'_2O contains no interior point of the triangle $P_1P_2P_3$.

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.

7.5.2 Finding a subset of q

In Figure 7.10, 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

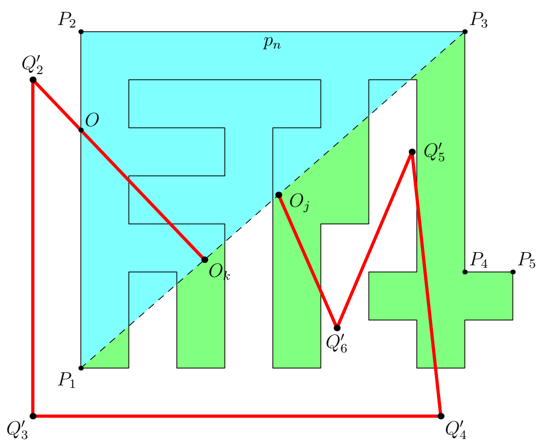


Figure 7.10: A chosen subset of q: $k = 2$ and $j = 6$

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.

Now there is nothing particularly informative about Veblen's last remark. Notice that the segment Q_1Q_2 in Figure 7.9 *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 properties are *invariants*.

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, once from the outside and once from the inside. More precisely, we just note that the interior of Q'_6O_k is outside the triangle, while the interior of Q'_2O_k is inside. To establish this, we note that the points inside the segment O_kO are all inside the triangle $P_1P_2P_3$.[3] Thus, by the Jordan Curve Theorem applied to triangles, all points of the polygonal segment $O\cdots Q'_2Q'_3Q'_4Q'_5Q'_6O_j$ must be outside the triangle $P_1P_2P_3$. 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 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.

7.5.3 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.

This is Veblen's conclusion, and the situation described is illustrated in Figure 7.11. The first step follows by the Jordan Curve Theorem applied to triangles: we know that $O_kQ'_2Q'_3Q'_4Q'_5O'_6O_j$ does not strictly[4] *cut* 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 $O_kQ'_2Q'_3Q'_4Q'_5O'_6O_j$ 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 $O_kQ'_2Q'_3Q'_4Q'_5O'_6O_j$ which is a subset of q. Presumably then, we need to find another subset of q which has the same properties as the original. And here we encounter the problem of detail mentioned at the end of §7.5.2:

[3]For Veblen, this is a matter of definition. See 9.3

[4]The use of phrases such as "strictly cut", among many other details, will be clarified in our verification. See Chapter 9.

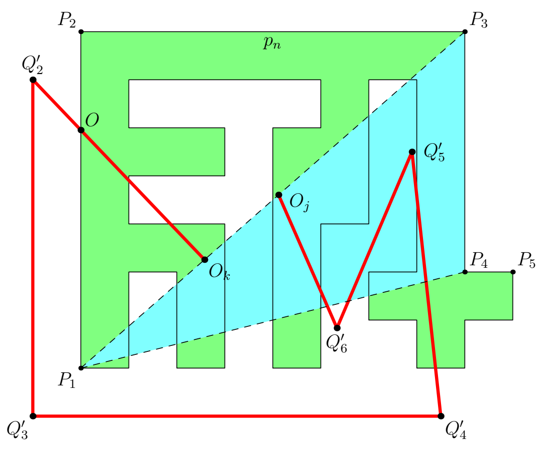


Figure 7.11: Intersections with $P_1P_3P_4$

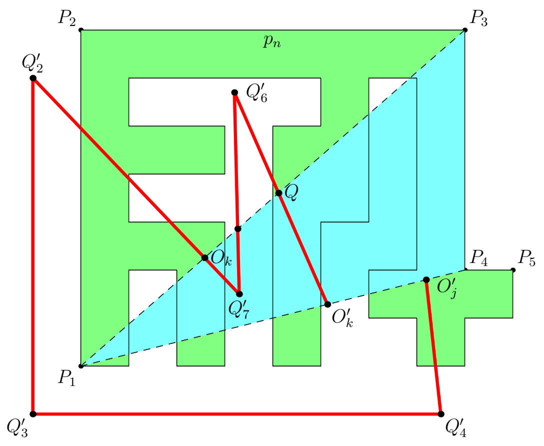


Figure 7.12: Desired subset of q?

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 7.12, namely $O_k'Q_6'Q_7'Q_2'Q_3'Q_4'O_j'$. This polygonal segment can be identified by starting at the point O_k and following Veblen's procedure as we did with the point O.

Remember that Veblen's next claim is that the polygonal segment $O_k'O_kQ_2'Q_3'Q_4'O_j'$ has a point inside and outside any triangle of which P_1P_3 is a side. Again, this is certainly true. But earlier, we justified it by noting that the polygonal segment touches P_1P_4 exactly twice, once from inside the triangle and once from outside. More formally, we can notice that the points inside the segment $O_k'Q$ lie inside the triangle $P_1P_3P_4$, while the points inside the segment $Q_4'O_j'$ lie outside the triangle. We could establish this earlier by relying on Veblen's assertion that the polygonal segment $O_k'O_kQ_2'Q_3'Q_4'O_j'$ crosses P_1P_3 *exactly once*, but this is no longer true. Instead, the required observation is that it crosses P_1P_3 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 crossings 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 crossings on a triangle's side. But as soon as we start to do *that*, we can immediately identify an argument which is conceptually much more straightforward. We give this argument in full in §9.1.

7.6 Conclusion

In this chapter, we have looked at several proofs of the Jordan Curve Theorem for simple polygons which might be applicable to Hilbert's two groups of axioms. Our first proof started with a recursive definition of an arbitrary polygon's interior and exterior. It explained how to add and subtract smaller polygons at each recursive step until we arrive at the base-case in triangles. The rest of the proof then shows that the recursive definition always yields two distinct regions which cover the plane, and such that any two points in the regions can be joined by a polygonal segment.

We then reviewed Veblen's 1904 proof of the theorem, from axioms equivalent to Hilbert's, and which, on the face of it, is far simpler to our first proof. However, it has been claimed elsewhere [19, 36] to be invalid. Based on a verification attempt, we believe we have identified the crucial problem, and have suggested that its fix leads immediately to a conceptually simple parity proof. In the next chapter, where we shall review our verification of the parity proof, we shall be able to constrast its conceptual simplicity with the complexity of its details.

With the verification, we have now unquestionably shown that despite the seeming theoretical poverty of Hilbert's first two groups, we have sufficient primitives and axioms to prove the Jordan Curve Theorem for polygons. We also hope that our analysis of the proof shows that Hilbert was mistaken to think the theorem could be obtained "without serious difficulty."

Chapter 8

The Formalisation of the Jordan Curve Theorem for Polygons

In this chapter, we will introduce our HOL Light formalisation of the polygonal Jordan Curve Theorem (*JCT* for short). This theorem appears as Theorem 9 in the 10th edition of Hilbert's *Foundations of Geometry*. The verification might stand on its own: the theorem's formulation does not appeal to earlier definitions or theorems, and of all the proofs we have considered, its verification is the most complicated by an order of magnitude. The theorem also serves as a useful case-study for the tools that we have developed up to this point: the linear reasoning tactic from Chapter 5 was used pervasively in verifying the theorem, becoming the crucial work-horse for solving and exploring complex linear ordering problems. We leveraged the theory of Half-Planes from Chapter 6 as the basis for key lemmas on two-dimensional ordering. And, as always, the discovery algebra described in Chapter 3 was needed throughout to handle incidence problems, to identify case-splits and suggest proof strategies based on discovered theorems.

8.1 Organisation

The verification is divided neatly into two parts, corresponding to the verification of two theorems:

1. Given two points in the plane of a simple polygon, any polygonal path between them must cross the simple polygon.

2. Given three points in the plane of a simple polygon, there is a polygonal path between two of them.

The formulation of these two lemmas is discussed in §8.3. This section, together with Appendix B.4, covers everything one needs to know about *what* we have verified, and can be read independently. The auxiliary concepts and associated lemmas which are needed to get the verification through will be covered in the next two chapters.

When we discuss the verifications, we will cover the same general topics. First, we shall give an overview of the theory structure, effectively amounting to a sketch proof of the theorem. From this, we shall give the formalisations of the 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 the concept 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 a 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 also hope to use these extracts to convince the reader that we have stayed faithful to the synthetic style of proof, even as they grow significantly in complexity. This, we suggest, is no mean feat. Consider that the verifications from Chapter 4 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.

We remind the reader that we use the word "formulate" to refer to the translating of intuitive ideas into a mathematically precise form, and the word "proof" to refer to mathematically rigorous arguments of formulated theorems. We reserve the word "formalise" for the translation of a mathematically precise statement into the language of higher-order logic, and the word "verification" for the HOL Light proof script which verifies the formalised theorem.

8.2 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 mathematic problems. The MIZAR [11] 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 the proof in HOL Light [21].

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 [14] 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.

8.3 Formulation

> DEFINITION. A set of segments AB, BC, CD, …, KL is called a *polygonal path* that connects the points A and L. Such a polygonal path 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 path*.
>
> In addition, for a polygonal path A, B, C, …, L, we shall call the segments AB, BC, CD, …, KL the *edges* of the polygonal path.

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 §2.2.2), Hilbert does not insist that the points involved here are distinct. Indeed, he adds an explicit distinctness clause when he defines polygons (see §8.3.2).

One thing which is clear to us is that Hilbert's notation is doing the heavy lifting in this definition. He needs that 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. Those 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 a computational metatheory. We saw this with Hilbert's Theorem 6, which we discussed at length in Chapter 5. Looking at Veblen's proof attempt, one would think the metatheoretical arguments needed for the Polygonal Jordan Curve Theorem are also mostly computational. This is not the case. Our verification shows that there is a *lot* more synthetic geometry involved in the proof than one would suspect from Veblen's attempt:

> 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.

Nevertheless, the notation is still doing heavy lifting, and so we should formalise the notation. 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 concrete list instances. Lists are available to us in the logic, ultimately being defined in terms of natural numbers, which as we saw in Chapter 5, are represented by geometric figures. We have something of a claim here that our lists are geometric rather than computational entities.

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 8.1, which 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. All the definitions of auxiliary functions, such as the list indexing function `el` are given in Appendix B.4.

With this function, 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`.

$$\begin{aligned}&\texttt{on\_polypath} : [\texttt{point}] \to \texttt{point} \to \texttt{bool}\\&\texttt{on\_polypath}\, Ps\, P \iff \\&\quad \texttt{mem}\, P\, Ps \lor \exists x\, y.\ \texttt{mem}\, (x,y)\ (\texttt{adjacent}\ Ps) \land \texttt{between}\ x\, P\, y.\end{aligned} \tag{8.1}$$

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 §6.1.2), 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. We shall call this relation "path-connectedness" and we shall say that two points satisying it are "path-connected."

$$\begin{aligned}&\texttt{path\_connected} : \texttt{plane} \to (\texttt{point} \to \texttt{bool}) \to \texttt{point} \to \texttt{point}\\&\texttt{path\_connected}\ \alpha\ figure\ P\ Q \iff\\&\quad \exists path.\ path \neq []\\&\qquad \land (\forall R.\ \texttt{mem}\ R\ path \implies \texttt{on\_plane}\ R\ \alpha)\\&\qquad \land \texttt{head}\ path = P \land \texttt{last}\ path = Q\\&\qquad \land \texttt{disjoint}\ (\texttt{on\_polypath}\ path)\ figure.\end{aligned}$$

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 tradeoffs 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.

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 path-connected with respect to a polygonal path Ps can be cleanly expressed by $\texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)$. This is obviously an equivalence relation.

8.3.1 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 these objects, we first proved some extra simplification rules for our list functions:

$$
\begin{aligned}
& xs \neq [\,] \wedge ys \neq [\,] \\
& \quad \Longrightarrow \texttt{adjacent}(xs +\!+ ys) \\
& \qquad\qquad = \texttt{adjacent}\ xs +\!+ (\texttt{last}\ xs,\ \texttt{hd}\ ys) +\!+ \texttt{adjacent}\ ys \\
& n + 1 < \texttt{length}\ xs \\
& \quad \Longrightarrow \texttt{el}\ n\ (\texttt{adjacent}\ xs) = (\texttt{el}\ n\ xs, \texttt{el}\ (n+1)\ xs) \\
& \texttt{length}\ xs = \texttt{length}\ ys \\
& \quad \Longrightarrow \texttt{reverse}\ (\texttt{zip}\ xs\ ys) = \texttt{zip}\ (\texttt{reverse}\ xs)\ (\texttt{reverse}\ ys) \\
& \texttt{butlast}\ (\texttt{reverse}\ xs) = \texttt{reverse}\ (\texttt{tail}\ xs) \\
& \texttt{tail}\ (\texttt{reverse}\ xs) = \texttt{reverse}\ (\texttt{butlast}\ xs).
\end{aligned}
$$

With these, we can prove that path-connectedness defines an equivalence relation on points in the plane which are not points of the figure. The constraint appears as an assumption on reflexivity.

Obviously, we will need these 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 path-connectedness.

$$
\begin{aligned}
& \texttt{on\_plane}\ P\ \alpha \wedge \neg \mathit{figure}\ P \implies \texttt{path\_connected}\ \alpha\ \mathit{figure}\ P\ P \\
& \texttt{path\_connected}\ \alpha\ \mathit{figure}\ P\ Q \implies \texttt{path\_connected}\ \alpha\ \mathit{figure}\ Q\ P \\
& \texttt{path\_connected}\ \alpha\ \mathit{figure}\ P\ Q \wedge \texttt{path\_connected}\ \alpha\ \mathit{figure}\ Q\ R \\
& \qquad \implies \texttt{path\_connected}\ \alpha\ \mathit{figure}\ P\ R.
\end{aligned}
$$

Note that for the rest of the chapter, 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. Our actual verification is conditional on information about planes, and we have restored the conditions in Appendix B.4, where it is clear they only serve to relativise all other objects to the same plane α. Had we been working from planar axioms, all of these terms would be absent (see §2.2.5 for some discussion).

8.3.2 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*.

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 on the formalisation side. As such, we shall only use them informally to explain parts of the verification. The important definition for the formalisation 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*.

Combining this with the definition of polygon gives us the following formalisation:

$$
\begin{aligned}
&\texttt{simple\_polygon}\ \alpha\ Ps \iff \\
&\qquad 3 \leq \texttt{length}\ Ps \\
&\qquad \wedge\, \texttt{head}\ ps = \texttt{last}\ Ps \\
&\qquad \wedge\, (\forall P.\ \texttt{mem}\ P\ Ps \implies \texttt{on\_plane}\ P\ \alpha) \\
&\qquad \wedge\, \texttt{pairwise}\ (\neq)\ (\texttt{butlast}\ Ps) \\
&\qquad \wedge\, \neg(\exists P\ Q\ X.\ \texttt{mem}\ X\ Ps \wedge \texttt{mem}\ (P,Q)\ (\texttt{adjacent}\ Ps) \wedge \texttt{between}\ P\ X\ Q) \\
&\qquad \wedge\, \texttt{pairwise}\ (\lambda(P,Q)\ (P',Q').\ \neg(\exists X.\texttt{between}\ P\ X\ P' \wedge \texttt{between}\ Q\ X\ Q').
\end{aligned}
\tag{8.2}
$$

In this definition, we have introduced the polygon's plane as a parameter, but we are aware of that this could be more elegantly 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 unweildly. 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 past Hilbert's constraints. The next two clauses are straightforward, and define *Ps* to be a polygon according to Hilbert's definitions. The real complexity is in the last three clauses which define the polygon as *simple*.

Given the unweildiness 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 (in fact, it almost *did*).

Because of these concerns, we must pay attention to 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` $\circ$ `lift2` 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) and even anti-symmetric. Consider the evaluation of `pairwise` $(<)$ $[1,2,3,4]$:

$$\begin{aligned} \texttt{pairwise}\ (<)\ [1,2,3,4,5] &= 1 < 2 < 3 < 4 \\ &\quad \wedge\ 2 < 3 < 4 \\ &\quad \wedge\ 3 < 4 \\ &= \top. \end{aligned}$$

[1]The number 4 would do just as well.

For now, inspection is the best way to inspire confidence in the definition, but there are two other small reassurances. Firstly, we have verifed a quick sanity check for triangles. Secondly, the definition does not appear until the final hurdles of our formal verification. Our verification of Veblen's first 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.

$$\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)$$
$$\implies \texttt{simple\_polygon}\ [A,B,C,A].$$

8.3.3 Goal Theorems

In §8.3.1, we said that we would understand the regions defined by the polygonal Jordan Curve Theorem as equivalence classes under path-connectedness. If we were to follow the style of Chapter 6, we would quotient an appropriate data-type under this relation and talk of regions as abstract objects.

We shall not, 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 proofs in this chapter show that there *was* a substantial 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 the Polygonal Jordan Curve Theorem is principally founded on the theory of half-planes.

Abstractly, then, we would express the Polygonal Jordan Curve Theorem by saying that a simple polygon separates the plane into exactly two regions. Talking instead in terms of the underlying representation and path-connectedness, we say firstly that there are at least two points in the plane and not on the polygon which are not 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 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 generalise and 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 and all points of B are path-connected 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 path-connectedness and path-connected regions from the first part. This condition appears in the second part, and can be obtained for the line a simply by setting $a = b$.

With the breakdown considered, we turn to the formalisation. Again, we must pay careful attention to the details. Our later verification demonstrates that these formalisations are provable, but we have left the matter of whether they are "too easily provable" to inspection.

$$\begin{aligned}
&\texttt{simple\_polygon}\ \alpha\ Ps\\
&\implies \exists P\ Q.\ \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha\\
&\qquad \wedge \neg\texttt{on\_polypath}\ Ps\ P \wedge \neg\texttt{on\_polypath}\ Ps\ Q\\
&\qquad \wedge \neg\texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q.
\end{aligned} \tag{8.3}$$

$$\begin{aligned}
&\texttt{simple\_polygon}\ \alpha\ Ps\\
&\wedge \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha \wedge \texttt{on\_plane}\ R\ \alpha\\
&\wedge \neg\texttt{on\_polypath}\ Ps\ P \wedge \neg\texttt{on\_polypath}\ Ps\ Q \wedge \neg\texttt{on\_polypath}\ Ps\ R\\
&\implies \texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q\\
&\qquad \vee \texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ R\\
&\qquad \vee \texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ Q\ R.
\end{aligned} \tag{8.4}$$

$$\begin{aligned}
&(\forall P. \texttt{on\_polypath}\ Ps\ P \implies \texttt{on\_plane}\ P\ \alpha)\\
&\implies \exists a.\ \forall P.\ \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ \alpha \wedge \neg\texttt{on\_polypath}\ P\ Ps.
\end{aligned} \tag{8.5}$$

$$\begin{aligned}
&(\forall P.\ \texttt{on\_polypath}\ Ps\ P \implies \texttt{on\_plane}\ P\ \alpha)\\
&(\forall P.\ \texttt{on\_line}\ P\ a \implies \texttt{on\_plane}\ P\ a)\\
&(\forall P.\ \texttt{on\_line}\ P\ b \implies \texttt{on\_plane}\ P\ b)\\
&\implies \texttt{on\_line}\ A\ a \vee \texttt{on\_line}\ A\ b \wedge \texttt{on\_line}\ B\ a \vee \texttt{on\_line}\ B\ b\\
&\implies \texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ A\ B.
\end{aligned} \tag{8.6}$$

Here, Theorems 8.3 and 8.4 formalise the fact that there are respectively at least two and at most two path-connected regions, while Theorems 8.5 and 8.6 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 8.3. We must assert that P and Q lie in the plane of the polygon and do not lie on the polygon. *Any* two points not satisfying these conditions are such that they cannot be path-connected. Without this side-condition, the theorem is trivial.

In this subsection, we have presented the verification goals for this chapter, namely Theorems 8.3 and 8.4. These theorems are also given in Appendix B.4, together with all definitions on which the formalisations depend. Any definitions and theorems we introduce from here on are scaffolding needed to support the verification. The verification is the subject of the next few chapters.

$$\texttt{adjacent} : [\alpha] \rightarrow [(\alpha,\alpha)]$$

$$\begin{aligned}
&\texttt{adjacent}\ [P_0,P_1,P_2,\ldots,P_n]\\
&\quad = \texttt{zip}\ (\texttt{butlast}\ [P_0,P_1,P_2,\ldots,P_n])\ (\texttt{tail}\ [P_0,P_1,P_{,}2,\ldots,P_n])\\
&\quad = \texttt{zip}\ \begin{array}{lccccccl} [& P_0, & P_1, & P_2, & \ldots, & P_{n-1} &] \\ [& P_1, & P_2, & P_3, & \ldots, & P_n &] \end{array}\\
&\quad = [(P_0,P_1),(P_1,P_2),(P_2,P_3),\ldots,(P_{n-1},P_n)]
\end{aligned}$$

$$\begin{aligned}
&\texttt{adjacent}\ [] = []\\
&\texttt{adjacent}\ (x : xs) = []\\
&\texttt{adjacent}\ (x : y : xs) = (x,y) : \texttt{adjacent}\ (y : xs).
\end{aligned}$$

Figure 8.1: Specifications for `adjacent`

Chapter 9

The Polygonal JCT: Part I

This chapter concerns the verification of Theorem 8.3 from Hilbert's axioms of ordered geometry. For this theorem, 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 similiar to Veblen's 1904 proof [77] which we described in detail in §7.5. We give the correct version of this proof now.

9.1 Sketch Proof

Consider the polygon Ps shown in Figure 9.1. We pick an arbitrary point X between P_1 and P_2 and then pick an arbitrary ray h through a point not on the line of AB. This ray intersects the polygon Ps a finite number of times, so we can find the point of intersection H closest to the point X. We then pick an arbitrary point A between X and H. For this point, we have that the segment AX does not intersect the polygon Ps. Finally, we consider the ray emanating from X in the other direction. Applying the same reasoning as above, we find a point B such that BX does not intersect Ps. We end up with a segment AB which intersects the polygon Ps exactly once between P_1 and P_2.

Now consider any polygonal path which connects A and B. Together with the segment AB, this yields another polygon Qs (possible 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

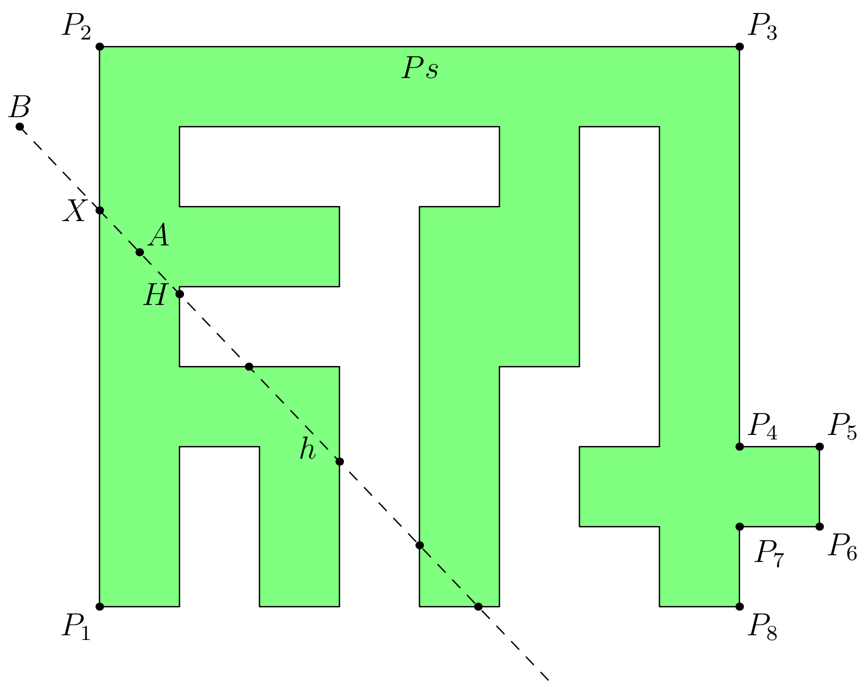


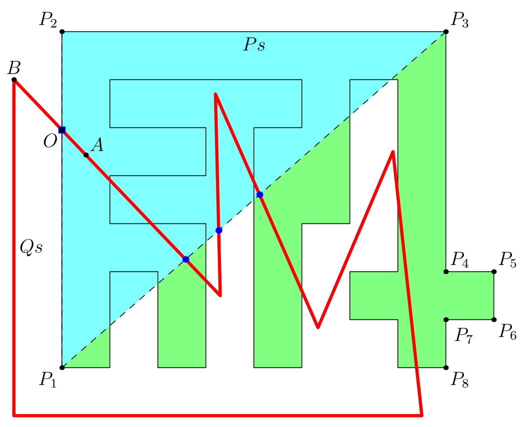
Figure 9.1: The Witnesses (A and B) for Theorem 8.3

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.

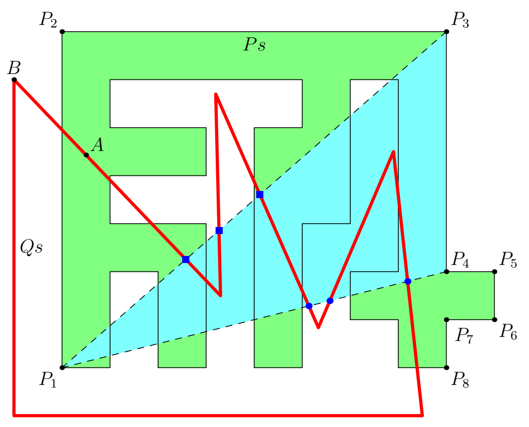
In Figures 9.2 and 9.3, we illustrate a run of this parity argument through the triangles $P_1P_2P_3$, $P_1P_3P_4$, $P_1P_5P_6$, $P_1P_7P_8$ (the steps for the triangles $P_1P_4P_5$, $P_1P_6P_7$ have been omitted for clarity). At the start of the proof, we assume that the polygon Qs crosses the edge 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 an *odd* number of times, to ensure that the total crossings are even, and indeed, there are 3 such crossings shown in Figure 9.2(a). Hence, we can continue with the triangle $P_1P_3P_4$.

We 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 in Figure 9.2(b). Again, we continue with the next triangle. Eventually, we shall find a

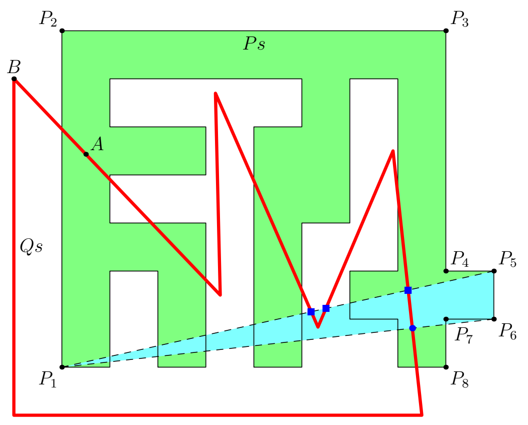


(a) Step 1: Four intersections

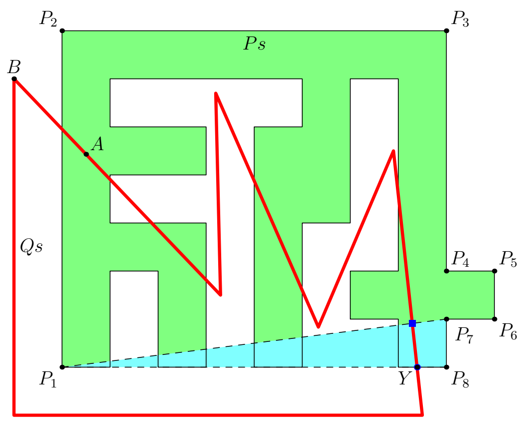


(b) Step 2: Six intersections

Figure 9.2: Parity Proof



(a) Step 4: Four intersections



(b) Step 6: Two intersections

Figure 9.3: Parity Proof (continued)

point of intersection with the polygon, shown as point Y in Figure 9.3.

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.

9.2 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.

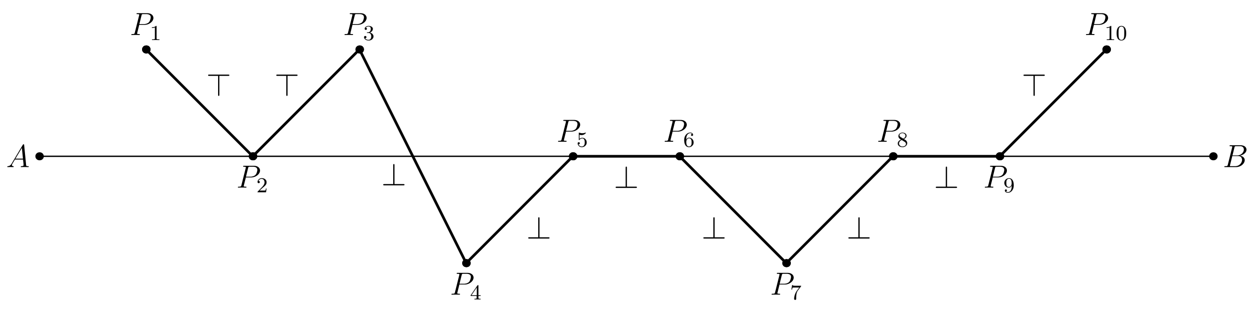


Figure 9.4: Assignment of Context on a Segment

9.2.1 Context

In our formulation, a polygon 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 crossings against a segment AB by reducing the problem to crossings of a single edge against AB. We can then define the crossings of the full polygonal path by summing the crossings at each of its edges.

This introduces an obvious 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 obviously, there is a crossing. 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 some larger polygonal path $P_{i-m}\ldots P_{i-1}P_iP_{n+1}\ldots\ldots P_{i+p}$ with $m, p > 0$.

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 `boolean` "context" variable Γ. We could, for instance, 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 difficult case, where P_{i+1} lies on the side AB itself, we can just propagate the preceeding context.

In Figure 9.4, we show a context value assigned in this way to the edges of $P_1\ldots P_{10}$. 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.

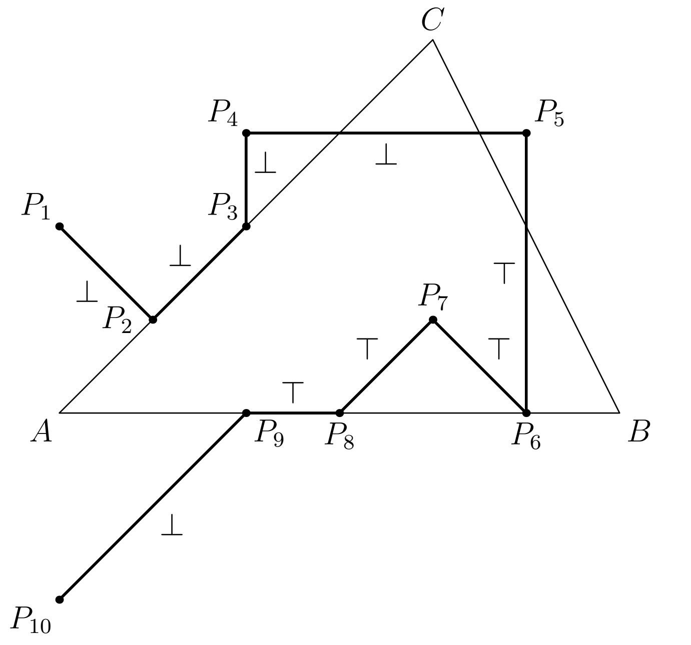


Figure 9.5: Assignment of Context in a Triangle

9.2.2 Combined Context for Triangles

We will be counting crossings on the edges of a triangle, which would require three context variables. Also, the assignment of $\top$ and $\perp$ to the sides of each edge of the triangle could not be arbitrary as it was in Figure 9.5. 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.

We chose a different approach. We declared the $\top$ side of each edge in a triangle to be the side containing the triangle's interior, and then we effectively combined the contexts by taking their conjunction. In other words, we used a single context variable which tracks whether a segment ends inside or outside the triangle.

In Figure 9.5, we show the assignment of context values to 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 $\perp$. If P_{i+1} lies inside the triangle, then the edge is assigned $\top$. If the point P_{i+1} lands on an edge, then we set the context as we did in the previous section.

With these context values, we can 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_iP_{i+1}. The only other crossings occur when P_i lies on the segment AB. Here, we count a crossing in two circumstances, which are consistent with the counting of crossings for the previous

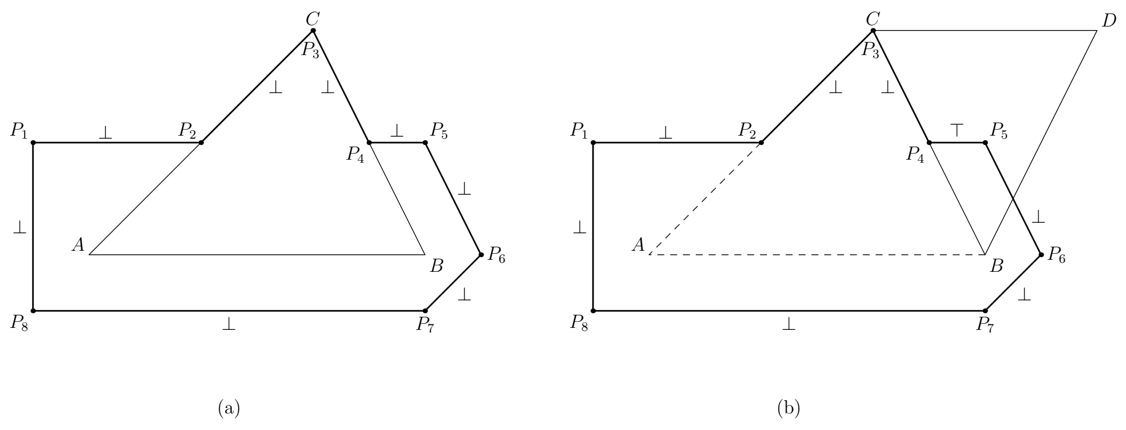


Figure 9.6: Context with Vertex Crossings

section:

- the context is $\perp$ and P_iP_{i+1} has a point in the interior of $\triangle ABC$ (and thus moves from outside to inside);
- the context is $\top$ and P_iP_{i+1} has a point in the exterior of $\triangle ABC$ (and thus moves from inside to outside).

Thus, in Figure 9.5, 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, in Section 9.5.3, we show 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. This theorem, together with several other key theorems, should fully clarify the intended semantics of a "crossing."

9.2.3 Avoiding Vertices

Our method of counting crossings of a polygonal path against the edges of a triangle breaks down when we allow the polygonal path to intersect the triangle's vertices. We have such a scenario in Figure 9.6(a). Here, we have drawn a *polygon* $P_1P_2P_3P_4P_5P_6P_7$ 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 9.6(b), we 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. This is 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.

9.2.4 Formalisation

We now introduce the formal functions with which we shall calculate the number of crossings against an edge of a triangle. The right-hand sides of these definitions are implementation details and are not particularly important. We provide them only to suggest the distance we have to cross to recover our intuitive and informal definition of a crossing.

Our first function computes the crossings at an edge of a triangle based on a context.

$$\texttt{crossing}\,(A,B,C)\,\Gamma\,P_i\,P_{i+1} = \begin{cases} 0, & \text{if } \texttt{between}\,A\,P\,B \land \texttt{between}\,A\,P_{i+1}\,B \\ 1, & \text{else if } \exists R.\texttt{between}\,P_i\,R\,P_{i+1} \land \texttt{between}\,A\,R\,B \\ 1, & \text{else if } \texttt{between}\,A\,P_i\,B \\ & \quad \land (\exists R.\texttt{between}\,P_i\,R\,P_{i+1} \\ & \qquad \land \texttt{in\_triangle}\,(A,B,C)\,R \iff \neg\Gamma) \\ 0, & \text{otherwise.} \end{cases} \tag{9.1}$$

We will briefly explain the function's arguments here. 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 Γ. The final two arguments are the endpoints of the polygonal path's edge against which we compute crossings.

Thus, to compute the crossings for the edges AC and BC, we just use the expressions $\texttt{crossing}\,(A,C,B)$ and $\texttt{crossing}\,(B,A,C)$, and to compute the total crossings of the segment P_iP_{i+1} on the triangle, we evaluate

$$\begin{aligned}\texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1}&\\ + \texttt{crossing}\ (B,A,C)\ \Gamma\ P_i\ P_{i+1}.&\end{aligned} \tag{9.2}$$

Our next function computes the context value for a segment P_iP_{i+1} based on the existing context. The arguments are the same, but here, the output does not depend on any particular ordering of the triple (A,B,C).

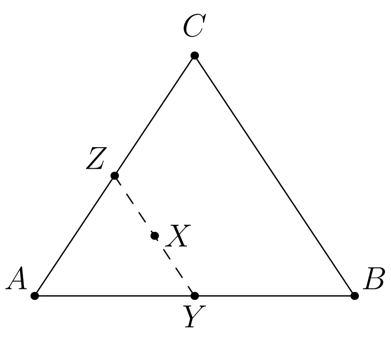
$$\begin{aligned}\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} \iff\ & \texttt{in\_triangle}\ (A,B,C)\ P_{i+1}\\ & \vee (\texttt{on\_triangle}\ (A,B,C)\ P_{i+1}\\ & \quad \wedge (\exists R.\texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{in\_triangle}\ (A,B,C)\ R)\\ & \quad\quad \vee \texttt{on\_triangle}\ (A,B,C)\ P_i \wedge \Gamma\end{aligned} \tag{9.3}$$

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 the this function still requires an initial context Γ. We shall show how to initialise it in §9.5.4.

$$\begin{aligned}&\texttt{polypath\_crossings}\ (A,B,C)\ \Gamma\ [\,] = 0\\ &\texttt{polypath\_crossings}\ (A,B,C)\ \Gamma\ ((P_i,P_{i+1}) : \mathit{segments})\\ &\quad = \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}\\ &\quad\quad + \texttt{polypath\_crossings}\ (A,B,C)\ (\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1})\ \mathit{segments}\end{aligned} \tag{9.4}$$

9.3 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 1904 thesis [77], 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 BC with X between Y and Z (see Figure 9.7). Here is another definition: the interior of $\triangle ABC$ is the set of all points on



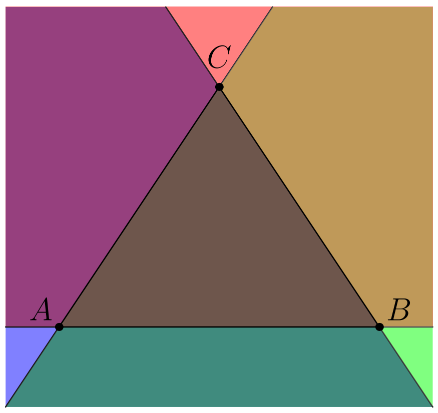


Figure 9.7: Two Definitions of a Triangle's Interior

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 interection of three half-planes. Each can be formalised as:

$$\begin{aligned}&\texttt{in\_triangle\_veblen}\ (A,B,C)\ P \iff\\ &\quad \exists X\ Y.\ \texttt{between}\ A\ X\ B \wedge \texttt{between}\ A\ Y\ C \wedge \texttt{between}\ X\ P\ Y.\end{aligned}$$

$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ P \iff\\ &\quad \exists hp\ hq\ hr.\ \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hp)\\ &\qquad \wedge \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hp)\\ &\qquad \wedge \texttt{on\_line}\ A\ (\texttt{line\_of\_half\_plane}\ hq)\\ &\qquad \wedge \texttt{on\_line}\ C\ (\texttt{line\_of\_half\_plane}\ hq)\\ &\qquad \wedge \texttt{on\_line}\ B\ (\texttt{line\_of\_half\_plane}\ hr)\\ &\qquad \wedge \texttt{on\_line}\ C\ (\texttt{line\_of\_half\_plane}\ hr)\\ &\qquad \wedge \texttt{on\_half\_plane}\ hp\ C \wedge \texttt{on\_half\_plane}\ hq\ B \wedge \texttt{on\_half\_plane}\ hr\ A\\ &\qquad \wedge \texttt{on\_half\_plane}\ hp\ P \wedge \texttt{on\_half\_plane}\ hq\ P \wedge \texttt{on\_half\_plane}\ hr\ P.\end{aligned} \tag{9.5}$$

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, Veblen's definition is first-order and more complicated to reason with. 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).

$$\begin{aligned}
&\texttt{on\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (A,C,B)\ P\\
&\wedge\,\texttt{on\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (B,A,C)\ P\\
&\wedge\,\texttt{in\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (A,C,B)\ P\\
&\wedge\,\texttt{in\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (B,A,C)\ P\\
&\wedge\,\texttt{outn\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (A,C,B)\ P\\
&\wedge\,\texttt{out\_triangle}\ (A,B,C)\ P \iff \texttt{on\_triangle}\ (B,A,C)\ P
\end{aligned} \tag{9.6}$$

That said, many of our early proofs 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 perservered instead with Veblen's definition, and tried to avoid reasoning about half-planes. We have some circumstantial against this: nowhere in our proof do we use the fact that our definition implies Veblen's. This indicates that we found the half-planes on the sides of a triangle to be more useful in proofs than the two point witnesses given in Veblen's definition. On the other hand, there were twelve places in our formalisation 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. These steps would have been unnecessary had we used Veblen's definition directly.

These two observations can be understood by again dividing geometric lemmas into those which introduce geometric entities and those which allow us to infer properties from a figure. As a theorem, Veblen's definition is weak for introducing entities, but useful for inferring properties of figures.

We make one final remark about our definition, which is important to keep in mind for some of the later verification. According to our definition, 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 meant we could implement a discoverer `add_in_triangle` to derive these non-collinear triples automatically and make them available to the `obviously` primitive.

We now briefly consider the formulation of a triangle's boundary and exterior. Since the boundary is just a polygonal path, it suffices to define:

$$\texttt{on\_triangle}\ (A,B,C)\ P \iff \texttt{on\_polypath}\ [A,B,C,A]\ P \tag{9.7}$$

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. These expressions 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 shall be relativising all of our theorems against a single plane, this does not matter.

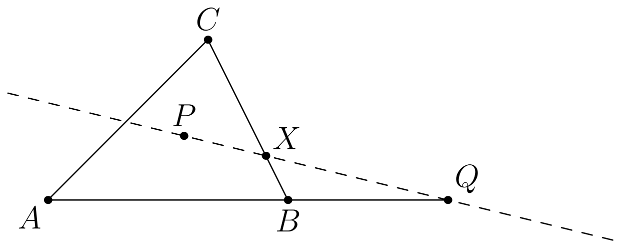
9.4 Some Prelimary Theorems

We recall the basic approach to synthetic axiomatic geometry as divided into two kinds of reasoning steps: one which introduces geometrical entities and one which identifies salient properties of the resulting figures. These properties may then allow us to introduce new configurations, and so on, until we have verified our goal theorem.

In the context of triangle interiors, we will need new rules to introduce points, and new rules to reason about geometric properties between interior and exterior points of these triangles. We have six theorems for this purpose, 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.

9.4.1 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



$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ P \wedge \texttt{between}\ A\ B\ Q \\ &\implies \exists X.\texttt{between}\ P\ X\ Q \wedge \texttt{between}\ B\ X\ C\end{aligned} \tag{9.9}$$

Figure 9.8: “Inner Pasch” for an interior point

between them which lies on the boundary. This therefore gives us two introduction rules, one for half-planes and one for triangles, and we use both frequently. The latter is even needed to prove our other introduction rule (9.10) in §9.4.4. In turn, this second introduction rule is crucial to the verification of the well-definedness of crossings at a triangle’s side (see §9.5.3).

$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ P \wedge \texttt{out\_triangle}\ (A,B,C)\ Q \\ &\qquad\implies \exists R.\texttt{on\_triangle}\ (A,B,C)\ R \wedge \texttt{between}\ P\ R\ Q\end{aligned} \tag{9.8}$$

We initially hoped the proof of this theorem 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 another point introduction lemma.

9.4.2 An “Inner Pasch” theorem

Initially, our only means to introduce points relative to triangles is 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 4.3 and 4.4. Our point introduction lemma can we thought of as a variation of Theorem 4.4. It says that, given an interior point P of a triangle ABC, and a point Qs outside the triangle on the ray AB, we can introduce the point at which the line P_iP_{i+1} intersects BC (we could then use the Outer Pasch Axiom to find the point at which P_iP_{i+1} intersects AC). See Figure 9.8.

The proof of this theorem illustrates some common patterns of reasoning with half-planes, and some of the pros and cons of our representation. The first half of the proof is shown in Figure 9.9. This block of script serves only to obtain the three half planes defining the triangle. The next three steps show that the lines of each half-plane 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.

That we need so many steps just to unfold our abstractions shows us the significant weaknesses in our representation. Fortunately, it is almost always the same steps which are needed. We shall omit such details, as well as any mention of planes from later proofs. We can only assure the reader that all steps in the verification are relativised to the same plane α.

The rest of the proof is shown in Figure 9.10. 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 the theorems B.10 and B.11. In contrast to the first part of the proof, this puts the use of half-planes in a much more positive light.

The basic strategy of the proof 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 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 translation that

```
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 $(D.1)$        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 $(9.5)$        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 $(I, 6), (B.6)$        17
...        18,19
```

Figure 9.9: Proof of “Inner Pasch” for an interior point (part 1)

```
obviously by_ncols hence on_plane $Q\ \alpha$        20
    $\neg$on_line $Q$ (line_of_half_plane $hr$)        21
    $\neg$on_half_plane $hr\ Q$ from 0,1,2,12,13,14,16 by $(I, 6), (II, 1), (B.9)$        22
consider $X$ such that on_line $X$ (line_of_half_plane $hr$) $\wedge$ between $P\ X\ Q$
 from 15,19,20,21,22 by $(B.9)$        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 $(II, 1), (B.10)$ from 7,23        24
hence $\neg$between $C\ B\ X$ from 5,6,17 by $(B.9)$        25
hence on_half_plane $hq\ Q$ from 8,10,16 by $(B.10)$
hence on_half_plane $hq\ X$ from 11,23 by $(B.11)$
hence $\neg$between $B\ C\ X \wedge B \neq X \wedge C \neq X$ from 5,9,10,18,24 by $(B.9), (B.7)$
obviously by_neqs qed from 0,12,13,23,25 by (Theorem 4)
```

Figure 9.10: Proof of “Inner Pasch” for an interior point (part 2)

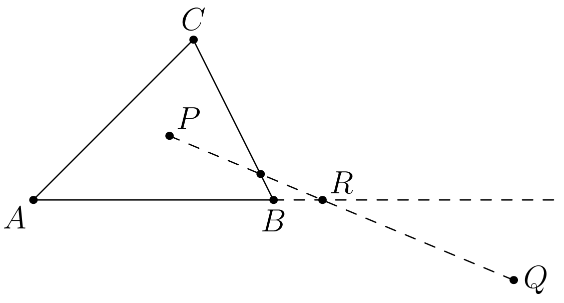


Figure 9.11: "Inner Pasch" to the Base Case

`on_half_plane` *hp*	becomes	on the same side of *AB* as *C* and *P*;
`on_half_plane` *hq*	becomes	on the same side of *AC* as *B* and *P*;
`on_half_plane` *hr*	becomes	on the same side of *BC* as *A* and *P*;
`line_of_half_plane` *hp*	becomes	the line *AB*;
`line_of_half_plane` *hq*	becomes	the line *AC*;
`line_of_half_plane` *hr*	becomes	the line *BC*.

With these translations in mind, we hope that the informal proof and its synthetic line of argument can be recovered. We can try to justify the translation by drawing an analogy between synthetic proofs and their accompanying diagrams. The diagram is strictly unnecessary, but can usually be easily recovered from prose, and it is often helpful to reconstruct it. Similarly, the informal argument can be easily recovered from the formal verification, where intuitive phrases such as "a ray emerging from the line" make things much easier to follow, even if such phrases do not point to interesting abstractions that would help the theorem prover.

9.4.3 From "Inner Pasch" to the Base Case

We can now give an informal proof of Theorem 9.8. 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 P_iP_{i+1} must intersect the line AB. If P_iP_{i+1} intersects the *segment* AB, we have found the required point on the triangle's boundary. Otherwise, there is a point R on the segment P_iP_{i+1} 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 Theorem 9.9 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

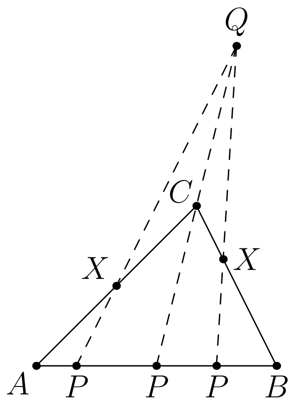


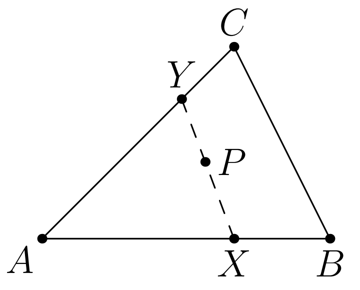
Figure 9.12: Another Point Introduction Rule

choice of AB and the point C. As Harrison has shown [26], 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" [1]: 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 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.

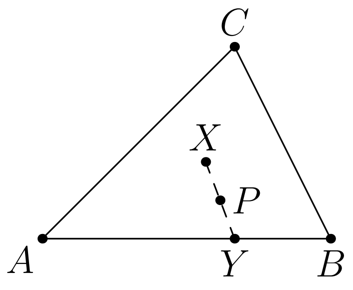
Instead, we formalised the above argument as a lemma, and then wrote an ad-hoc procedural script to manually apply the symmetries. Admittedly, a triangle only admits at most six symmetries and the procedural boilerplate is hardly a bottle-neck compared to our use of MESON in declarative proofs. Still, this represents a somewhat inelegant approach in our verification, one we would like to investigate in future work.

9.4.4 Additional Theorems

We have one more rule to introduce points. Here, we suppose that we have a point P on the edge of a triangle ABC and a point Q on the same side of AB as C. In this case, the segment P_iP_{i+1} must intersect the polygonal path $[A,B,C]$ at a point X (see Figure 9.12).



(a) Points between the sides of a triangle are interior (9.11)



(b) Points between an interior point and a side are interior (9.12)

Figure 9.13: Triangle Interior Lemmas

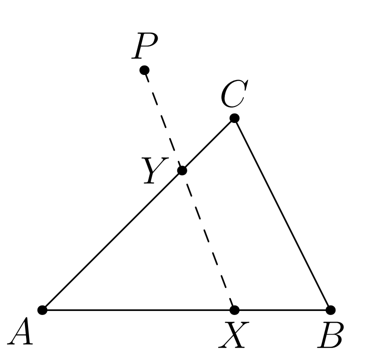
$$\begin{aligned}
&\texttt{between}\,A\;P\;B\\
&\wedge\,\texttt{on\_line}\,A\;(\texttt{line\_of\_half\_plane}\;hp)\wedge\texttt{on\_line}\,B\;(\texttt{line\_of\_half\_plane}\;hp)\\
&\wedge\,\texttt{on\_half\_plane}\;C\;hp\wedge\texttt{on\_half\_plane}\;Q\;hp\\
&\wedge\,\texttt{out\_triangle}\;(A,B,C)\;Q\implies\exists X.\texttt{between}\;P\;X\;Q\wedge\texttt{on\_polypath}\;[A,B,C]\;X
\end{aligned}\tag{9.10}$$

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 §9.5). 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.

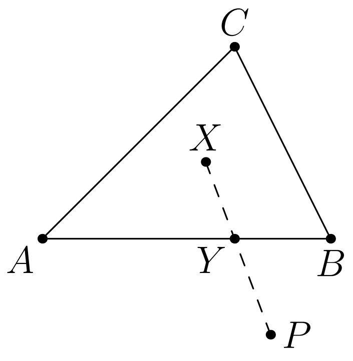
We give diagrams and a short description for each theorem in Figures 9.13 and 9.14. These theorems all have reasonably clear synthetic proofs, and together require 82 proof steps. Roughly two fifths of these proof steps are assisted by our incidence discoverer via the `obviously` and `clearly` primitives.

Note that Theorem 9.11 is one direction of the equivalence between our definition of triangles and Veblen's (see §9.3).

$$\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\;A\;a\wedge\texttt{on\_line}\;B\;a\wedge\texttt{on\_line}\;C\;a)\\
&\wedge\,\texttt{between}\;A\;X\;B\wedge(\texttt{between}\;A\;Y\;C\vee C=Y)\\
&\implies\texttt{between}\;X\;P\;Y\implies\texttt{in\_triangle}\;(A,B,C)\;P
\end{aligned}\tag{9.11}$$



(a) A ray through an opposite side leaves the triangle (9.13)



(b) A ray from inside the triangle to a side emerges outside the triangle (9.14)

Figure 9.14: Triangle Exterior Lemmas

$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)\ X \wedge \texttt{on\_triangle}\ (A,B,C)\ Y\\ &\implies \texttt{between}\ X\ P\ Y \implies \texttt{in\_triangle}\ (A,B,C)\ P\end{aligned} \tag{9.12}$$

$$\begin{aligned}&\neg(\exists a. \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\ &\wedge \texttt{on\_triangle}\ (A,B,C)\ X \wedge \texttt{on\_triangle}\ (A,B,C)\ Y\\ &\implies \texttt{between}\ X\ Y\ P \implies \texttt{out\_triangle}\ (A,B,C)\ P\end{aligned} \tag{9.13}$$

$$\begin{aligned}&\texttt{in\_triangle}\ (A,B,C)X \wedge \texttt{on\_triangle}\ (A,B,C)\ Y\\ &\implies \texttt{between}\ X\ Y\ P \implies \texttt{out\_triangle}\ (A,B,C)\ P\end{aligned} \tag{9.14}$$

9.5 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.

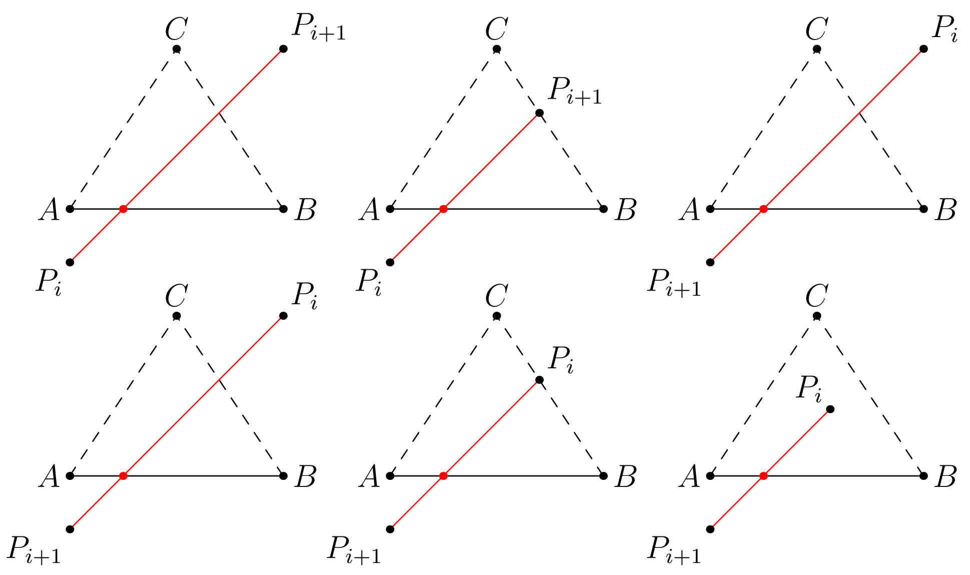


Figure 9.15: Cases of Crossings

9.5.1 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 (6.6) that we discussed in §6.2.2. We do not need any messy case-splits, only a short piece of procedural script to eliminate WLOG assumptions. We end up with this:

$$\begin{aligned} &\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ &\Longrightarrow \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1} \\ &\qquad + \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} \leq 2 \end{aligned} \tag{9.15}$$

The case-splits only appear for our next major stepping stone: a 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 9.15.

$$
\begin{aligned}
&\wedge \neg(\exists a. \texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\
&\wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ A \wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ B \wedge \neg\texttt{on\_polypath}\ [P_i, P_{i+1}]\ C\\
&\wedge (\neg\texttt{on\_triangle}\ (A,B,C)\ P_i \implies (\texttt{in\_triangle}\ (A,B,C)\ P_i \iff \Gamma))\\
\implies &\left(\begin{aligned}&\texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} + \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1}\\ &\qquad + \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} = 1 \iff \Gamma = \neg\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}\end{aligned}\right)
\end{aligned} \tag{9.16}
$$

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 §9.2.3.

The rest of the theorem then clarifies both the idea behind a crossing and the idea behind the context variable Γ. The third hypothesis says that if P_i is not a point on the triangle, then the context variable Γ must correctly track whether P_i is inside or outside the triangle. The conclusion then 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 above theorem hypothesises that Γ tracks whether the point P_i of the segment P_iP_{i+1} is inside or outside a triangle. Since P 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 vertices.

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.

$$
\begin{aligned}
&\neg\texttt{on\_triangle}\ (A,B,C)\ P_{i+1}\\
&\qquad \implies (\texttt{in\_triangle}\ (A,B,C)\ P_{i+1} \iff \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1}).
\end{aligned} \tag{9.17}
$$

9.5.2 Some Verification

We will not bore the reader with the details of the verification. Instead, we will give an extract of a specific case, showing how in these proof we are still relying on our

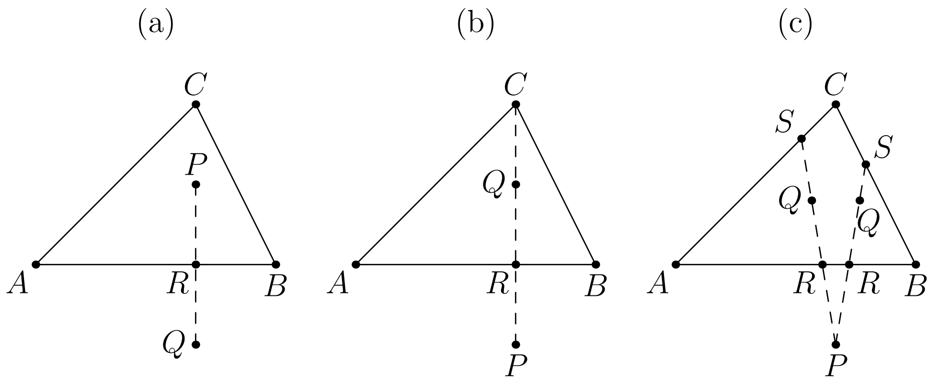


Figure 9.16: Main Case Split

discovery algebra from Chapter 3, our linear ordering tactic from Chapter 5, and how we leverage our lemmas for this section, and thus avoid having to deal directly with half-planes.

The case we shall consider is, in effect, the base case for IH3 from our first sketch proof (see §7.4.2). We are going to say that if a segment P_iP_{i+1} crosses a triangle ABC exactly once at the point R, then its endpoints are in different regions with respect to the triangle.

$$
\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\
&\wedge \texttt{between}\ P_i\ R\ P_{i+1} \wedge \texttt{between}\ A\ R\ B\\
&\quad \wedge \texttt{crossing}\ (A,C,B)\ \Gamma\ P_i\ P_{i+1} = 0 \wedge \texttt{crossing}\ (B,C,A)\ \Gamma\ P_i\ P_{i+1} = 0\\
&\wedge \neg\texttt{on\_polypath}\ [P_i,P_{i+1}]\ A \wedge \neg\texttt{on\_polypath}\ [P_i,P_{i+1}]\ B\\
&\quad \wedge \neg\texttt{on\_polypath}\ [P_i,P_{i+1}]\ C\\
&\wedge \neg\texttt{on\_triangle}\ (A,B,C)\ P_i \wedge \neg\texttt{on\_triangle}\ (A,B,C)\ P_{i+1}\\
&\qquad \implies (\texttt{in\_triangle}\ (A,B,C)\ P_i \iff \texttt{out\_triangle}\ (A,B,C)\ P_{i+1}) \quad (9.18)
\end{aligned}
$$

We divide the proof into the three cases shown in Figure 9.16. In case (a), we have assumed that P_i is interior. It then follows immediately from Theorem 9.14 that P_{i+1} is exterior. In case (b), we have assumed that P_i is interior and that P_i and P_{i+1} are in line with the vertex C. In this case, we just apply Theorem 9.12. Finally, in case (c), we have assumed that P 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 the line of P_iP_{i+1} and thus obtain a point S either on AC or BC. It then follows from Theorem 9.11 that P_{i+1} is interior.

Actually, things are not *quite* so simple for cases (b) and (c). In order to apply Theorem 9.14 in case (b), we first have to prove that P_i 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 (9.12) 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 proof 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 (9.12) 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 P_iP_{i+1} does not intersect either A or B. Here, we proceed by contradiction. 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. It was not immediately obvious to us what this necessary information was.

Assuming that A, P_i and P_{i+1} lie a line, the linear reasoning tactic will first infer that A, B, P_i, P_{i+1} and R are all collinear. Since P_{i+1} does not lie on the triangle, this would force A and B to lie between P_i and P_{i+1}, which is impossible because P_iP_{i+1} does not intersect any of the triangle's vertices. This suggests that we should prove the following facts, and indeed, once armed with them, our tactic can solve the goal by reasoning across the five points:

$$\begin{aligned}&A \neq P_i \wedge A \neq P_{i+1} \wedge B \neq P_i \wedge B \neq P_{i+1}\\ &\qquad \wedge \neg\texttt{between}\ P_i\ A\ P_{i+1} \wedge \neg\texttt{between}\ P_i\ B\ P_{i+1} \wedge \neg\texttt{between}\ A\ P_i\ B \qquad (9.19)\end{aligned}$$

When we expand this one case to deal with the situation where P starts on AB and the context is used, we have a proof which runs to 70 steps. We are putting Mizar Light through its paces, and for the most part, it copes very well with the complexity.

In the places where the prover struggled, it was usually simple enough to apply the `using` and `tactics` combinators to inject pieces of procedural proof into the script. In doing this, we generally tried to respect the aims of declarative proof. 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.

For instance, the automatic prover often struggles to apply complex conditionals with many hypotheses. For these situations, we add an initial `MATCH_MP_TAC` in a `using` clause. This leaves the default prover `MESON` with just the theorem's assumptions as goals. This tactic still keeps things largely declarative, since we name the theorem we are matching against and we do not modify any assumptions.

In other places, we found it helpful to introduce a function `mutual_simp` which takes a list of theorems and simplifies each with respect to the others. This function is used to process the justifying theorems of a declarative step before they are handed to `MESON`. The need for such processing comes from the fact that a declarative proof is based on accumulating theorems in a proof context. These theorems are expected to influence one another without any destructive modification (consider, as a simple example, an equation inferred late in a verification which could rewrite all previous assumptions). It is often helpful to have the theorems modify one another as they are applied at a step, using a function such as `mutual_simp`, so that they are more useful to `MESON`.

Finally, for theorems with hypotheses such as `crossing` $(A,C,B)\ X\ P_i\ P_{i+1} = 1$, we need to do some work in unfolding the mess of case-splits defined by the equation. 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 proof. With the cases converted, the `assume` steps allow us to make more meaningful assumptions, as in the proof extract in Figure 9.17.

9.5.3 Crossings are Well-defined

In our sketch proof, 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 prove 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.

```
theorem ¬(∃a.on_line A a ∧ on_line B a ∧ on_line C a)
    ∧ crossing (A,B,C) X P_i P_{i+1} = 1
    ∧ crossing (A,C,B) X P_i P_{i+1} = 1
    ⟹ crossing (B,C,A) X P_i P_{i+1} = 0
assume ¬(∃a.on_line A a ∧ on_line B a ∧ on_line C a)
tactics unfold_crossing_tac
assume between A P_i B ∨ ∃R.between P_i R P_{i+1} ∧ between A R B
```

Figure 9.17: Unfolding Crossings

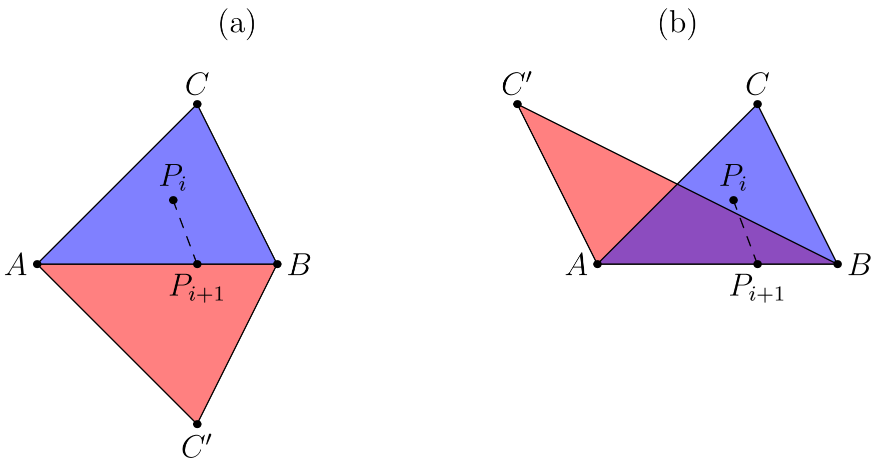


Figure 9.18: Triangles Sharing an Edge

There is an obvious case-split here, shown in Figure 9.18. If C and C' are on opposite sides of AB as in (a), 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 move from the exterior of ABC to the interior. On the other hand, if C and C' are on the same side of AB as in case (b), then the triangle interiors will overlap. Here, as we cross the edge AB, we enter or leave the interiors of both triangles together.

$$\begin{aligned}
&\texttt{on\_line}\,A\ (\texttt{line\_of\_half\_plane}\ hp) \land \texttt{on\_line}\,B\ (\texttt{line\_of\_half\_plane}\ hp)\\
&\land \texttt{on\_half\_plane}\ hp\ C \land \texttt{on\_half\_plane}\ hp\ C' \land \texttt{between}\,A\ P_i\ B\\
&\qquad\Longrightarrow \begin{pmatrix} (\exists R.\texttt{between}\ P_i\ R\ P_{i+1} \land \texttt{in\_triangle}\ (A,B,C)\ R) \\ \iff (\exists R.\texttt{between}\ P_i\ R\ P_{i+1} \land \texttt{in\_triangle}\ (A,B,C')\ R) \end{pmatrix}
\end{aligned} \tag{9.20}$$

$$\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\,A\ a \land \texttt{on\_line}\,B\ a \land \texttt{on\_line}\,C'\ a)\\
&\land \texttt{on\_polypath}\ [P_i;P_{i+1}]\ A \land \texttt{on\_polypath}\ [P_i;P_{i+1}]\ B\\
&\land \texttt{on\_line}\,A\ (\texttt{line\_of\_half\_plane}\ hp) \land \texttt{on\_line}\,B\ (\texttt{line\_of\_half\_plane}\ hp)\\
&\land \texttt{on\_half\_plane}\ hp\ C \land \neg\texttt{on\_half\_plane}\ hp\ C'\\
&\land \texttt{between}\,A\ P_i\ B \land \neg\texttt{between}\,A\ P_{i+1}\ B\\
&\qquad\Longrightarrow \begin{pmatrix} (\exists R.\texttt{between}\ P_i\ R\ P_{i+1} \land \texttt{in\_triangle}\ (A,B,C)\ R) \\ \iff \neg\exists R.\texttt{between}\ P_i\ R\ P_{i+1} \land \neg\texttt{in\_triangle}\ (A,B,C')\ R \end{pmatrix}
\end{aligned} \tag{9.21}$$

These two theorems are proven by reasoning about half-planes and, in both cases, applying Theorem 9.10. The assumptions on half-planes in Theorem 9.20 require that the points C and C' lie on the same side of AB. In Theorem 9.21, they require that C and C' lie on opposite sides. Theorem 9.21 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 $\texttt{between}\,A\ P_i\ B$. This reflects the fact that the case-split is only pertinent when we come to update and make use of the context variable Γ. This variable is only needed 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 set to say that we were last inside or outside the triangle. When it emerges from AB, the context variable must be correctly applied to say whether the edge P_iP_{i+1}

counts as a crossing. These matters can be formally clarified by the corollaries in Figures 9.19 and 9.20 (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 he hit the segment AB, and we expect them to be the same when they leave this segment. When C and C' are on opposite sides, we expect the context values to be opposite when we hit AB, and when we expect them to stay opposite when they leave this segment.

Now that 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), we 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 have abstracted away the vertex C. The mechanics of this will become clear in our final proof in §9.6.1

First, we must consider how we initialise Γ.

9.5.4 Initialising the Context

Recall that the context variable Γ is needed when we want to count crossings of the edge P_iP_{i+1} of a polygonal path and an edge AB for a triangle ABC. If one of the endpoints P_i and P_{i+1} is on the boundary of ABC, then the number of crossings 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 decide on this initial context?

Sometimes, the answer is straightforward. If the endpoint P_{i+1} lies in the interior of the triangle, then the value of the context for P_iP_{i+1} is $\top$. If P_{i+1} lies in the exterior of ABC, then the value is $\bot$. These give us our solutions for the initial context, and it would be convenient if we could rely on this simple case.

There is some hope here, since any closed polygon which crosses a triangle must have a vertex which is not on the triangle. So perhaps all we have to do is arrange things so that we only start computing a crossing for a polygon at a vertex which lies either inside or outside the triangle. This will require that we have some way to rotate a polygon's vertex list, but we will show how to do this in §10.5.1.

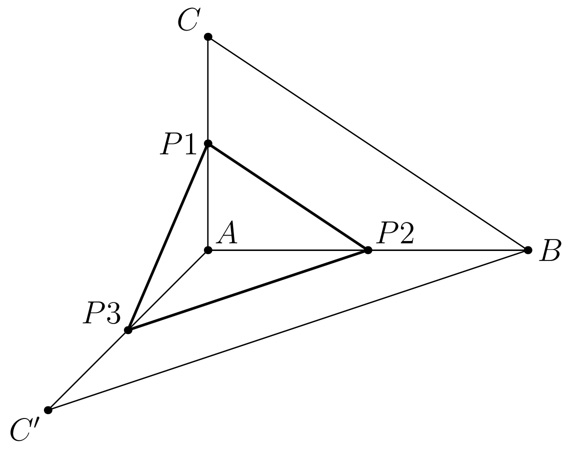


Figure 9.21: No Well-defined Initial Context

Unfortunately, this simple method breaks the well-definedness considered in the last section. In Figure 9.21, we have two triangles ABC and ABC' sharing an edge AB, each crossed by the polygonal path $P_1P_2P_3$. As we claimed, this polygonal path has a vertex off the boundary of each triangle (P_3 for $\triangle ABC$ and P_1 for $\triangle ABC'$), but there is no vertex off both boundaries.

Thus, we simply cannot avoid the question of how to initialise the context when a polygonal path starts on the triangle's boundary.

Here is an alternative. We must first understand the context as telling us whether or not we were recently inside the triangle. We then realise that the initial value of the context for a polygonal path should be sensibly related to the value of the context at the polygonal path's final edge. We can compute this final value with a recursive function:

$$\begin{aligned}
&\Gamma_{final}\ (A,B,C)\ \Gamma\ [] = \Gamma \\
&\Gamma_{final}\ (A,B,C)\ \Gamma\ ((P_i,P_{i+1})\ :\ segments) = \\
&\qquad \Gamma_{final}\ (A,B,C)\ (\Gamma_{next}\ (A,B,C)\ \Gamma\ \ P_i\ P_{i+1})\ segments
\end{aligned} \tag{9.26}$$

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 should then give us a suitable starting context, and it will appear in our final specification in the next section.

9.5.5 The Specification of Crossings

We are at last able to recover the intuitive idea behind crossings from the mess of case-analyses and implementation detail of the previous sections. In Figure 9.22 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 our verification of Lemma 1 in §9.6.1.

The first, Theorem 9.27 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 9.28 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 §9.2.3. Under these conditions, the total number of crossings against the three sides is always even.

Theorem 9.29 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 set the initial context appropriately using the context Γ' given in the conclusion, but since Theorem 9.28 holds for arbitrary choices of Γ, we can ignore this constraint when we apply the two theorems.

The upshot of Theorem 9.29 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 view plays out in our verification.

9.6 Verifying the Sketch Proof

In this section, we shall review our verification of the parity proof that we sketched in §9.1. There are two ideas we find particularly noteworthy here: firstly, there are interesting details in the formalisation relating to our use of the theorems of the previous section. But more importantly, we can show that the first half of the Polygonal Jordan Curve Theorem arises as a corollary to a major theorem that we give at the end of this section. Unlike the Polygonal Jordan Curve Theorem, this theorem makes no reference to *simple polygons*. In fact, the proof leads to what we regard as a far more elegant theorem about arbitrary polygonal paths, one which does not hinge on such complex definitions as the Polygonal Jordan Curve Theorem.

9.6.1 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:

$$\begin{aligned}
&\texttt{length}\ Qs \geq 2 \wedge \texttt{head}\ Qs = \texttt{last}\ Qs \wedge P_1 \neq P_2 \\
&\wedge (\forall C. \neg(\exists a. \texttt{on\_line}\ P_1\ a \wedge \texttt{on\_line}\ P_2\ a \wedge \texttt{on\_line}\ C\ a) \\
&\qquad\qquad \Longrightarrow \exists \Gamma. \texttt{odd}\ (\texttt{polypath\_crossings}\ (P_1, P_2, C) \\
&\qquad\qquad\qquad\qquad (\Gamma_{final}\ (P_1, P_2, C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs))) \\
&\quad \Longrightarrow \exists X. \texttt{on\_polypath}\ ([B] ++ Ps ++ [P_1])\ X \wedge \texttt{on\_polypath}\ Qs\ X
\end{aligned} \tag{9.30}$$

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 AB 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 `crossing`. To do this, 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.

Rather than use structural induction, we proceed by induction on the length of Ps. This is because we have virtually no constraints on the vertices of the two polygons: we do not even assume that the vertices in Ps are distinct. In particular, it might be that Ps has a prefix made up of copies of P_2. These need to be broken off if we are to apply our inductive hypothesis, and what we end up with is some shorter list of unspecified structure.

We take this opportunity to introduce the `break` function, which splits a list in two at the first point at which a predicate evaluates to true. We have defined this function rather clumsily using an auxiliary function and an accumulator, but what emerges after a few procedural proofs is an elegant and complete specification which is trivial to apply in formal proofs:

$$\texttt{all}\ (\lambda x.\neg(p\,x))\ (\texttt{fst}\ (\texttt{break}\ p\,xs))$$
$$\texttt{exists}\ p\,xs \to \texttt{snd}\ (\texttt{break}\ p\,xs) \neq [] \wedge p(\texttt{head}\ (\texttt{snd}\ (\texttt{break}\ p\,xs)))$$
$$\texttt{fst}\ (\texttt{break}\ p\,xs) + +\texttt{snd}\ (\texttt{break}\ p\,xs) = xs$$

Using the expression break $(\lambda P.P_2 \neq P)\ Ps$, we can break off all the repeated copies of P_2. If we are left with an empty list, then the tail of our polygon is effectively just $[P_2, P_1]$, a single segment against which we know by assumption that there is a crossing. Thus, in this case, we just apply Theorem 9.27 to find the required point of intersection.

If the break does not leave us with an empty list, then we consider the first vertex P_3 in the list which is distinct from P_2. Our aim will be to show that there is an odd number of crossings on P_1P_3, after which we can apply the inductive hypothesis. According to our treatment of this idea from §9.5.3, our goal is formalised as

$$\begin{aligned}&\forall C.\neg(\exists a.\texttt{on\_line}\ P_1\ a \wedge \texttt{on\_line}\ P_3\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad\Longrightarrow \exists\Gamma.\texttt{odd}\ (\texttt{polypath\_crossings}\ (P_1,P_3,C)\\ &\qquad\qquad (\Gamma_{final}\ (P_1,P_3,C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs))\end{aligned}$$

In proving this, we get to assume that the polygon Qs does not intersect the segment P_2P_3, since otherwise we are done. In Mizar Light, the assumption is made quite literally:

$$\boxed{\texttt{assume}\ \neg(\exists X.\texttt{on\_polypath}\ [B,C]\ X \wedge \texttt{on\_polypath}\ Qs\ X)}$$

There is actually yet another 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 9.29, which we give in Appendix C.3. 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

$$\begin{aligned}&\forall C.\neg(\exists a.\texttt{on\_line}\ P_1\ a \wedge \texttt{on\_line}\ P_2\ a \wedge \texttt{on\_line}\ C\ a)\\ &\quad\Longrightarrow \exists\Gamma.\texttt{odd}\ (\texttt{polypath\_crossings}\ (P_1,P_2,C)\\ &\qquad\qquad (\Gamma_{final}\ (P_1,P_2,C)\ \Gamma\ (\texttt{adjacent}\ Qs))\ (\texttt{adjacent}\ Qs)))\end{aligned}$$

[1]For this inference, we use our linear reasoning tactic

It should now be clear how the quantifiers in this expression 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 9.27 and 9.28, 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 (9.29).

9.6.2 A Theorem of Polygonal Paths

Theorem 9.30 is all we need to reach the claim that a simple polygon divides the plane into two regions. It is interesting to note, however, that we have not mentioned path connecteness in this theorem, nor have we supposed that the polygons concerned are simple. In fact, such an assumption is overkill even for the main theorem we intend to prove in this chapter. This suggests there are other interesting theorems to be had from the ideas considered thus far, and we shall consider one in this section.

This little detour allows us to present the main results of this section in a way which does not rely on bulky formalisations about *crossings*. We have come this far by building a large tower of abstractions, complicated and unweildy definitions, and theorems containing far too many hypotheses. The pay-off from this sort of verification is a result which throws out such rickety scaffolding and brings us to a neat and easily grasped theorem.

In a fairly short proof (42 steps), we apply Theorem 9.30 to obtain a highly symmetric theorem concerning arbitrary intersecting polygons.

$$\begin{aligned}
&\neg(\exists a.\mathtt{on\_line}\ P_1\ a \wedge \mathtt{on\_line}\ P_2\ a \wedge \mathtt{on\_line}\ Q_1\ a)\\
&\wedge \neg(\exists a.\mathtt{on\_line}\ Q_1\ a \wedge \mathtt{on\_line}\ Q_2\ a \wedge \mathtt{on\_line}\ P_1\ a)\\
&\wedge \mathtt{between}\ P_1\ X\ P_2 \wedge \mathtt{between}\ Q_1\ X\ Q_2\\
&\wedge P_1 = \mathtt{last}\ Ps \wedge Q_1 = \mathtt{last}\ Qs\\
&\implies \exists Y.\mathtt{on\_polypath}\ (P_2 : Ps)\ Y \wedge \mathtt{on\_polypath}\ (Q_1 : Q_2 : Qs)\ Y\\
&\qquad\qquad \vee \mathtt{on\_polypath}\ (P_1 : P_2 : Ps)\ Y \wedge \mathtt{on\_polypath}\ (Q_2 : Qs)\ Y
\end{aligned} \tag{9.31}$$

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.

$$(\texttt{between}\ A\ P_i\ B \implies \Gamma = \Gamma') \implies \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \texttt{crossing}\ (A,B,C')\ \Gamma'\ P_i\ P_{i+1} \tag{9.22}$$

$$\neg\texttt{between}\ A\ P_i\ B \wedge \texttt{between}\ A\ P_{i+1}\ B \implies \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} \tag{9.23}$$

Figure 9.19: Well-definedness Theorems when C and C' are on the same side of AB

$$(\texttt{between}\ A\ P_i\ B \implies \Gamma = \neg\Gamma') \implies \texttt{crossing}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \texttt{crossing}\ (A,B,C')\ \Gamma'\ P_i\ P_{i+1} \tag{9.24}$$

$$\neg\texttt{between}\ A\ P_i\ B \wedge \texttt{between}\ A\ P_{i+1}\ B \implies \Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} = \neg\Gamma_{next}\ (A,B,C)\ \Gamma\ P_i\ P_{i+1} \tag{9.25}$$

Figure 9.20: Well-definedness Theorems when C and C' are on opposite sides of AB

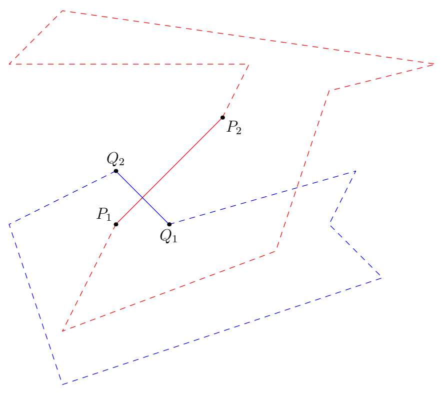


Figure 9.23: Arbitrary Intersecting Polygons

$$\texttt{polypath\_crossings}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Ps) > 0 \\ \implies \exists.\texttt{on\_polypath}\ Ps\ Q \wedge\ \texttt{between}\ A\ Q\ B \tag{9.27}$$

$$\begin{aligned} & Qs = [P] ++ Ps ++ [P] \\ & \wedge\, \Gamma_{initial} = \Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs) \\ & \wedge\, \neg\texttt{on\_polypath}\ Qs\ A \wedge \neg\texttt{on\_polypath}\ Qs\ B \wedge \neg\texttt{on\_polypath}\ Qs\ C \\ & \wedge\, \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ & \quad \implies \texttt{even} \begin{pmatrix} \texttt{polypath\_crossings}\ (A,B,C)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \\ +\ \texttt{polypath\_crossings}\ (A,C,B)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \\ +\ \texttt{polypath\_crossings}\ (B,C,A)\ \Gamma_{initial}\ (\texttt{adjacent}\ Qs) \end{pmatrix} \end{aligned} \tag{9.28}$$

$$\begin{aligned} & Qs = [P] ++ Ps ++ [P] \\ & \wedge\, \neg\texttt{on\_polypath}\ Qs\ A \wedge \neg\texttt{on\_polypath}\ Qs\ B \\ & \implies \forall C\ C'.\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\ & \qquad\qquad \wedge\, \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C'\ a) \\ & \quad \implies \exists\Gamma'.\texttt{polypath\_crossings}\ (A,B,C)\ (\Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs)) \\ & \qquad\qquad\qquad (\texttt{adjacent}\ Qs) \\ & \qquad\quad = \texttt{polypath\_crossings}\ (A,B,C')\ (\Gamma_{final}\ (A,B,C')\ \Gamma'\ (\texttt{adjacent}\ Qs)) \\ & \qquad\qquad\qquad (\texttt{adjacent}\ Qs) \end{aligned} \tag{9.29}$$

Figure 9.22: Final Specification of Crossings

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 9.23.

9.7 The Plane Divides into at Least Two Regions

We can now prove 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 §9.1, and the formalisation 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 formalisation, this step is handled by a "ray-casting" theorem which we discuss in §10.3.1.

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, of course, 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*.

$$\begin{aligned}
&\texttt{simple\_polygon}\ \alpha\ Ps\\
&\implies \exists P\ Q.\ \texttt{on\_plane}\ P\ \alpha \wedge \texttt{on\_plane}\ Q\ \alpha\\
&\qquad \wedge \neg\texttt{on\_polypath}\ Ps\ P \wedge \neg\texttt{on\_polypath}\ Ps\ Q\\
&\qquad \wedge \neg\texttt{path\_connected}\ \alpha\ (\texttt{on\_polypath}\ Ps)\ P\ Q.
\end{aligned} \tag{8.3}$$

Chapter 10

The Polygonal JCT: Part II

In this chapter, we shall discuss the second half of the Jordan Curve Theorem for polygons based on the axioms of Hilbert's ordered geometry, and our verification of this theorem in HOL Light. In this half of the theorem, 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 8, 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, and it has a pleasing visual interpretation. Unlike the "crossings" of the last chapter, the basic concepts we appeal to can be realised faithfully in the low-level details of the formalisation, 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 §10.1, we shall give an overview of the proof structure, effectively amounting to a sketch proof of the theorem. In §10.2, we will look more closely at some of the basic ideas behind the sketch proof, and formalise them in higher-order logic. Then, in §10.3, we shall cover the key lemmas that support our basic formalised concepts. As in the last chapter, we shall divide these lemmas between 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 what we hope are interesting proofs, demonstrating how faithfully we can formalise the intuitive synthetic arguments.

10.1 Sketch Proof

The basic intuition behind the proof is similar to the ones presented by Veblen [77] and Feigl [18]. We follow Veblen's proof the most closely. Contary to Guggenheimer's [19] claim that Veblen's proof only holds for convex polygons, we believe it is 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. Each edge of the polygon can then be thought of as a maze *wall*.

Our basic goal is to get the three players "next to" the same wall. Then we just need to find a path between whichever of the two players are on the same side of that wall. 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 [75]. In Tverberg's proof, we just need to consider a sufficiently small region around the walls of the maze (an "offset curve"), which we know to be 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 wall of the maze is to assert that all three have line-of-sight to that wall. 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.

In Figure 10.1, we show players *Red*, *Black* and *Blue* starting at various points of a maze. Players *Red* and *Black* are inside the maze, while *Blue* is on the outside. Here, we depict the paths they follow as they traverse the maze so that they have line-of-sight to the wall P_iP_{i+1}. Since *Red* and *Black* are 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

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

and suppose that the witness C in the conclusion is always chosen so that B is half-way

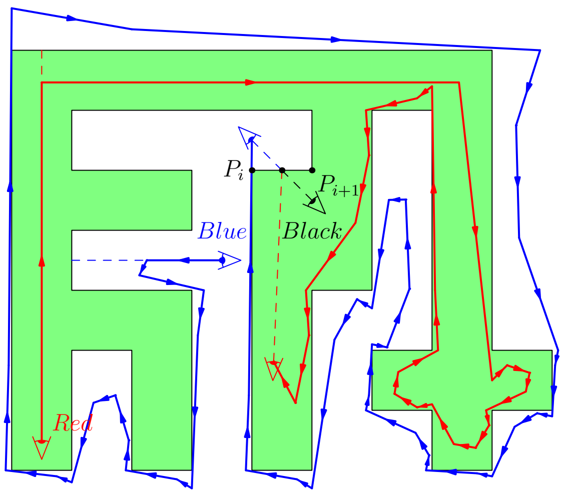


Figure 10.1: Navigating a Maze

between A and C. In this case, the paths sketched in Figure 10.1 are precisely those that we witness in our formal verification.

10.2 Formulation and Formalisation

Compared to the last chapter, where we introduce the complex idea of a crossing, the basic ideas needed in the proof for this chapter are relatively straightforward. Firstly, given a simple polygon Ps, we will say that a point X has line-of-sight to be a point X' if there is no point of Ps which lies strictly between X and X'. In diagrams such as Figure 10.1, we shall always depict the point X as an eye pointed at X', with a dashed line indicating the line-of-sight. We shall also 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 *wall* P_iP_{i+1}. The situation is formalised as

$$\begin{gathered}\neg\texttt{on\_polypath}\ Ps\ X \wedge \texttt{between}\ P_i\ X'\ P_{i+1} \\ \wedge\neg\exists Z.\texttt{between}\ X\ Z\ X' \wedge \texttt{on\_polypath}\ Ps\ Z\end{gathered}$$

Our formalised proof breaks down into three parts. Firstly, we must show how every point not on a simple polygon has a line-of-sight to some wall of the maze. Next, we must show how, if a point X has line-of-sight to a wall P_iP_{i+1}, then there is a polygonal path to a point Y which has line-of-sight to the next wall $P_{i+1}P_{i+2}$. As such, for any wall and any point X, there is a polygonal path from X to another point which has line-of-sight to that wall. Finally, we must show that if two players have line-of-sight to the same wall, and lie on the same side of that wall, then there is a polygonal path between them. We shall describe the informal proofs and verifications of each part in §10.4.

First, we consider the crucial supporting lemmas.

10.3 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.

10.3.1 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 §9.7), but it makes more sense to explain it in this section where we are self-consciously using 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 is the only major theorem since Hilbert’s 5.17 which relies almost exclusively on linear reasoning. We also found it proved to be particularly tricky. As with our proof of Theorem 9.18 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 ray-casting proof was deciding which hypotheses were needed.

Finding the necessary hypotheses 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

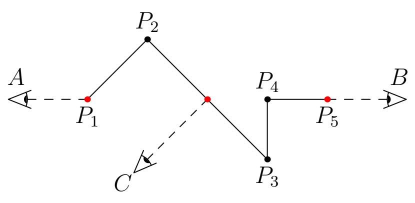


Figure 10.2: Ray-casting

additional facts to feed the tactic and then retry.

In a few places, we found that what we were missing was a case-split. 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 basic proof 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 10.2. 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 proofs, 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.

$$\begin{aligned}
\vdash & \neg \texttt{on\_polypath}\ Ps\ X \wedge \texttt{on\_polypath}\ Ps\ P \\
& \implies \exists Y.\texttt{on\_polypath}\ Ps\ Y \wedge \texttt{between}\ X\ Y\ P \vee X = P) \\
& \qquad \wedge \neg(\exists Q.\texttt{between}\ X\ Q\ Y \wedge \texttt{on\_polypath}\ Ps\ R).
\end{aligned} \tag{10.1}$$

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 §9.7 of the last chapter. There, we said that the final part of the proof 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*.

10.3.2 Squeeze

The most powerful theorem in our development towards the Jordan Curve Theorem 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. Again, what counts as a narrow gap in the abstract world of ordered geometry is only determined by the three-place *betweenness* relation. Our ability to navigate these gaps must therefore reduce to just the basic axioms governing this relation. 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, the theorem tells us that if we have a polygonal path $[A,B,C]$ which is not intersected by the polygonal path Ps, then there is a point A' between A and C such that Ps intersects AC in at most one place. We can understand this in the context of finding diagonals of simple polygons, as is described in Figure §7.4.1. In fact, this was the original motivation for *squeeze*, and we have retained its conclusion in a form which is still fit for this purpose. It thus remains an important component of a verification for our sketch proof in §7.4. In §10.3.3, we will show how to interpret it in terms of lines-of-sight.

$$\begin{aligned}
&\vdash \neg\texttt{on\_polypath}\ \mathit{path}\ B \\
&\quad \wedge \neg(\exists X.\texttt{between}\ A\ X\ B \wedge \texttt{on\_polypath}\ \mathit{path}\ X) \\
&\quad \wedge \neg(\exists X.\texttt{between}\ B\ X\ C \wedge \texttt{on\_polypath}\ \mathit{path}\ X) \\
&\quad\quad \implies \exists A'.\texttt{between}\ A\ A'\ B \wedge \neg\exists X.\texttt{in\_triangle}\ (A',B,C) \wedge \texttt{on\_polypath}\ \mathit{path}\ X.
\end{aligned} \tag{10.2}$$

In Figure 10.3, we show how *squeeze* would be used 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 $P_1P_2P_3\ldots P_{10}$.

There are two points to note about our conclusion. Firstly, we make a stronger claim than merely 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 diagonal with the point C. Secondly, even though the witness A' is defined in our proof to be such that a vertex P_8 of Ps lies on $A'B$, we have not provided

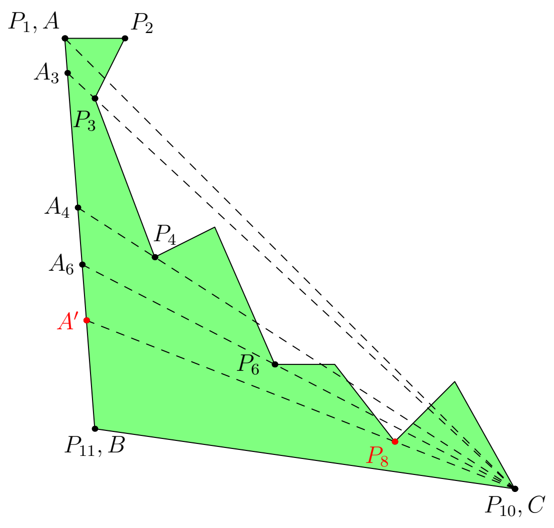


Figure 10.3: Squeezing a Diagonal

this additional information in the conclusion. Such information would be needed if our diagonal is to split the polygon into two simpler polygons, as needed for our first sketch proof in §7.4. We *have* verified this theorem with the additional information provided. However, for the purposes of the current proof, we present only the version with the simpler conclusion.

The mention of triangle interiors in the conclusion reflects the way the proof recursively verifies that $A'C$ really is a diagonal. Conceptually, the proof starts with the triangle ABC, and finds the first vertex in Ps which lies inside this triangle. In the case shown, this would be the vertex P_3. We then 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 (§9.5). Rather than go into the details of these, we shall focus on the more illuminating proof that $A'BC$ contains no *vertex* of Ps. This only boils down to two more lemmas for triangle interiors.

10.3.2.1 Another "Inner Pasch" Rule

Our first supporting theorem is a point introduction rule which allows us to find the intersection points A_3, A_4, A_6 and A' in Figure 10.3. This theorem is similar in spirit

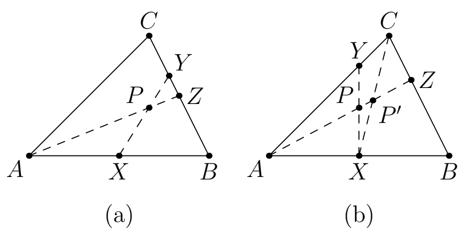


Figure 10.4: Drawing a line from a Vertex to the Opposite Side

to the Pasch axioms (4.4, 4.3) and its variant for triangle interiors (9.9). It has a very succinct formalisation, but a non-trivial verification:

$$\vdash \texttt{in\_triangle}\,(A,B,C)\;P \implies \exists X.\texttt{between}\;B\;X\;C \land \texttt{between}\;A\;P\;X. \qquad (10.3)$$

The proof is the most interesting of these point introduction rule proofs. Unlike the proof of Theorem 10.3, 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 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 10.4, then we apply (II, 4) to the triangle BXY and the line AP to find a point 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 (9.12).

In case (b) of Figure 10.4, we find that P is now an interior point according to Veblen's definition, in virtue 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.

10.3.2.2 Subtriangles

In our proof of Theorem 10.2, we consider a sequence of triangles $\triangle ABC$, $\triangle A_3BC$, $\triangle A_4BC$, $\triangle A_6BC$ and $\triangle A'BC$. Each of these 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 in the exterior. 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.

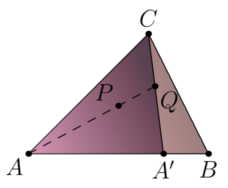


Figure 10.5: Any interior point P of $\triangle ABC$ is an interior point of $\triangle AB'C$.

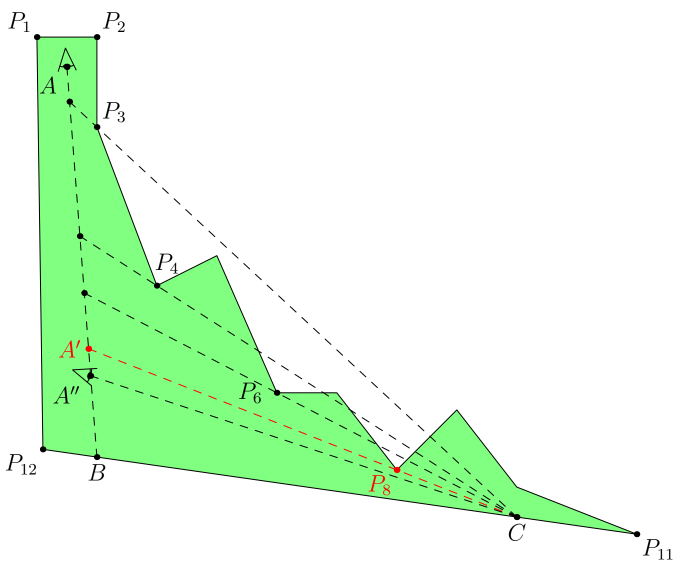


Figure 10.7: Obtaining line-of-sight to C

The proof, given in Figure 10.6, is extremely short and readable, making use of a number of lemmas we have previously considered, including Theorem 10.3 from the last subsection. With this theorem, we can put a point Q on the line $A'C$, where we know it must be interior to $\triangle ABC$ on the basis of Theorem 9.11. It then follows by Theorem 9.12 that P must also be interior.

10.3.3 Obtaining lines-of-sight via Squeeze

As we suggested earlier, Theorem 10.2 is quite powerful, and allows us to do more than obtain diagonals. 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 proofs of this chapter.

10.3.3.1 Moving to a new line-of-sight

Consider Figure 10.7. Here, we are not interested in diagonals, so we have slightly modified and relabelled the diagram from Figure 10.3. Now we want to interpret the point A is our player's starting location, with line-of-sight to a point B on the wall

$P_{11}P_{12}$ of a simple polygon. The goal is to obtain line-of-sight to a given point C on that same wall.

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 10.2 and taking *path* to be the fragment of the polygon $P_{12}P_1P_2\dots P_{11}$. Let us think about how to fulfill 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 path $P_{11}P_1P_2\dots P_{11}$, because AB is supposed to be a line-of-sight. We also know that BC does not intersect $P_{11}P_{12}$, because we are assuming *simple* polygons: the polygon should not self-intersect the edge $P_{11}P_{12}$, and this edge contains BC. We will still the possibility that $P_{11}P_1P_2\dots P_{11}$ intersects the lone point C, since as we shall see in §10.4.1, we will sometimes need to obtain line-of-sight to a vertex, and thus set $C = P_{11}$.

Applying Theorem 10.2 is not quite enough. We now have a point A' which *almost* has line-of-sight to C. But that point A' was originally intended to give us diagonals of polygons intersecting at least one vertex, and that vertex (P_8) is still in our way. We can fix this using 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 (10.2), a segment which is a line-of-sight to the point C. We capture all of this in a corollary:

$$\begin{aligned}
&\vdash \neg \texttt{on\_polypath}\ \mathit{path}\ B \\
&\quad \wedge \neg(\exists X.\texttt{between}\ A\ X\ B \wedge \texttt{on\_polypath}\ \mathit{path}\ X) \\
&\quad \wedge \neg(\exists X.\texttt{between}\ B\ X\ C \wedge \texttt{on\_polypath}\ \mathit{path}\ X) \\
&\quad \implies \exists A'.\texttt{between}\ A\ A'\ B \wedge \neg\exists X.\texttt{between}\ A'\ X\ C \wedge \texttt{on\_polypath}\ \mathit{path}\ X.
\end{aligned} \tag{10.4}$$

Now because AB was a line-of-sight, we know that AA'' is part of a polygonal path which does not intersect $P_1P_2\dots P_{11}P_{12}P_1$. 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 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 wall.

The next use of squeeze will be just as critical for our proof.

`theorem in_triangle` (A,A',C) $P \wedge$ `between` $A\ A'\ B \implies$ `in_triangle` (A,B,C) P

`assume in_triangle` (A,A',C) $P \wedge$ `between` $A\ A'\ B$ 0

`so consider` Q `such that between` $A'\ Q\ C \wedge$ `between` $A\ P\ Q$ 1

`obviously (by_ncols` $\circ$ `add_in_triangle` $\circ$ `conjuncts)`

  `hence in_triangle` (A,B,C) Q `by` (9.6), (9.11), (II, 1) `from` 0

`qed from` 1 `by` (9.7), (9.12)

Figure 10.6: “Subtriangles”

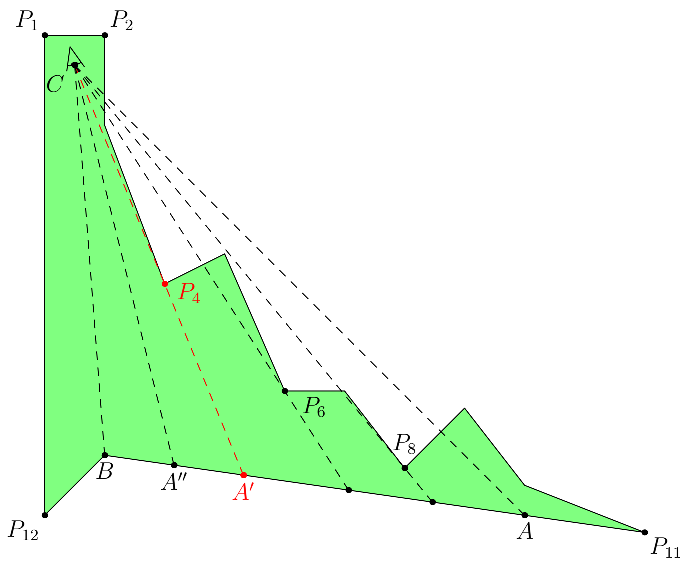


Figure 10.8: Obtaining line-of-sight to the interior of AB

10.3.3.2 Rotating to a new line-of-sight

In Figure 10.8, 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 proof, 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 wall). So we want our player to rotate their line-of-sight slightly.

Therefore, we apply Theorem 10.2 by reversing the order of A, B and C. The polygon 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 10.2 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 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'. This 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 prove 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 this version of Axiom 10.4. We shall always be assuming that, when we rotate our line-of-sight, the point P_{12} will be known to be on the opposite side of BC as the point P_{11}, as shown.

To summarise, we can say that the first use of Theorem 10.4 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.

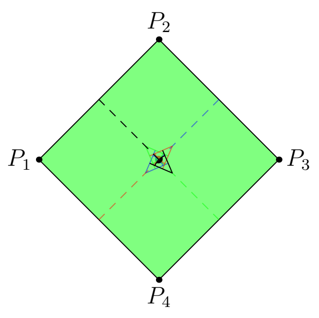


Figure 10.9: Wall-to-wall in a convex polygon

10.4 Wall-to-Wall

Since a simple polygon is just a list of adjacent edges, we can navigate every single wall just by moving between one wall and the next in the adjacency list. In Section 10.2, we explained that the movement from a wall P_iP_{i+1} to a wall $P_{i+1}P_{i+2}$ 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* wall, as in Figure 10.9. The salient fact to notice is that, in a convex polygon, interior points are on the same side of every wall as every other vertex.

Generalising this, we can say that two walls 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} (and equivalently, on the same side of $P_{i+1}P_{i+2}$ as P_i). Otherwise, we can say that the walls appear "locally concave". These provide our two cases for how we navigate the walls of a simple polygon.

10.4.1 Locally Convex Walls

In this subsection, we shall deal with the case of locally convex walls. Here, we shall assume we have a polygon $P_1P_2P_3\ldots P_n$, and we shall assume that we are moving from wall P_1P_2 to wall P_2P_3 (we can use the polygon rotations described in §10.5.1 to consider the other pairs of walls).

In Figure 10.10, 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 apply-

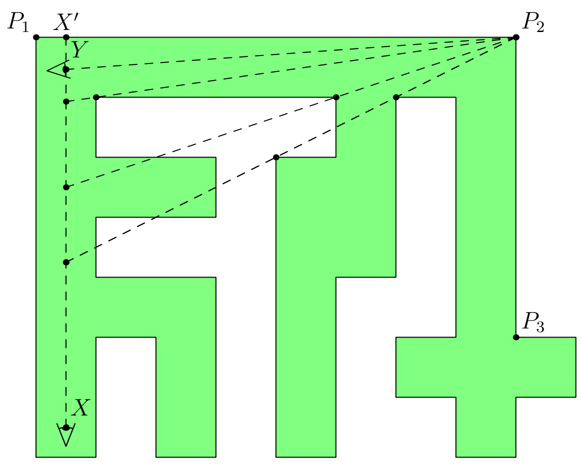


Figure 10.10: Wall-to-wall in a locally convex polygon

ing Theorem 10.4 as described in §10.3.3.1, seeking a line-of-sight with the vertex P_2.

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. We typically started procedurally. First, we would assert our goal without any justification:

$$\texttt{consider}\ Y\ \texttt{such that between}\ X\ Y\ X'$$
$$\wedge\neg\exists Z.\texttt{between}\ Y\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 : P_3 : Ps)\ Z.$$

Mizar Light's default prover rapidly fails to prove this as a logical truth, and as a lucky consequence of the implementation, we are dumped back at the interpreter as if we had split off the asserted claim as a new subgoal.

We then use `MATCH_MP` to retrieve our hypotheses as a big conjunction, and then split each off into its own subgoal. We then tackle these one at a time. Afterwards, we can gather up all the dependent assumptions and prove the original goal in one step. We just need a bit of tactic script to help `MESON`, but as usual, we make sure not to destructively modify the goal state, and we reference our justifying theorem (10.4). The step is still largely declarative.

$$\texttt{obviously by\_incidence so consider}\ Y\ \texttt{such that between}\ X\ Y\ X'$$
$$\wedge\forall Z.\texttt{between}\ Y\ Z\ P_2 \implies \neg\texttt{on\_polypath}\ (P_2 : P_3 : Ps)\ Y$$
$$\texttt{from}\ \ldots\ \texttt{by on\_polypath}, (\text{II}, 2)$$
$$\texttt{using K (MATCH\_MP\_TAC}\ (10.4)\ \texttt{THEN EXISTS\_TAC}\ \alpha) \qquad 16, 17$$

This done, we have our desired path XY. We know that this path does not intersect

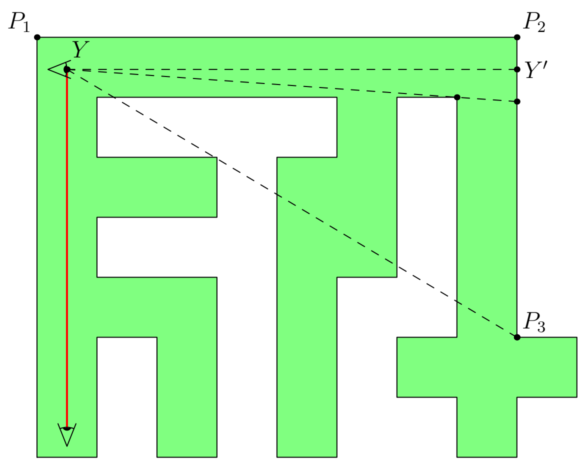


Figure 10.11: Wall-to-wall in a locally convex polygon (continued)

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 10.11, using Theorem 10.4 as described in §10.3.3.2.

Of course, 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 (10.4) only tells us that it does not intersect the fragment of the polygon $P_3P_4 \ldots P_1$. Our automated incidence tool tells us further that it 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. Since P_1P_2 and P_2P_3 are locally convex, we know that Y and P_3 are on the same side of P_1P_2. Now Y' lies on the ray P_2P_3, and so must also be on the same side of P_1P_2. And then, any point on $Y'Y$ must be on the same side of P_1P_2, and thus, cannot *intersect* P_1P_2.

The verification shown in Figure 10.12 matches this argument's structure almost exactly. We start by assuming there is some point Z on YY' which intersects the polygonal path $[P_1, P_2, P_3]$, and proceed by contradiction (we have removed some of the references to earlier steps and inserted elipses for readability).

10.4.2 Locally Concave Walls

When the walls 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 wall.

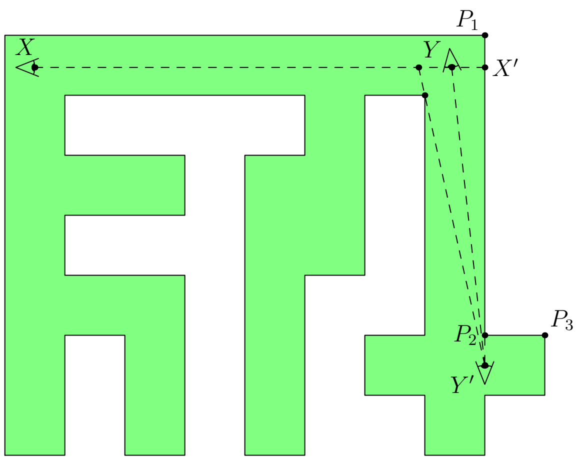


Figure 10.13: Wall-to-wall in a locally concave polygon

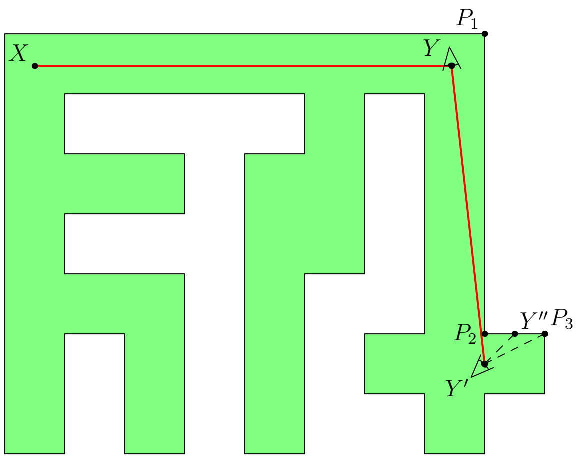


Figure 10.14: Wall-to-wall in a locally concave polygon (continued)

Before we apply Theorem 10.4, we will pick an appropriate target line-of-sight. This will be the point Y' shown in Figure 10.13. This point is "just off" the vertex P_2, and can be found by ray-casting (Theorem 10.1) from P_2 in the direction $\overrightarrow{P_1P_2}$. We then use Theorem 10.4 as described in §10.3.3.1, 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 10.4 as described in §10.3.3.2. We thus have the required path and line-of-sight to Y'' as shown in Figure 10.14.

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

$Y'Y''$ does not intersect the polygon.

The way these segments were obtained via Theorem 10.4 guarantees that they do not intersect the fragment $P_3P_4 \ldots P_n$, so that leaves us to consider whether either of the 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 wall P_1P_2. We only need to use half-planes to show that it does not intersect the wall P_2P_3, using the same sort of tidy declarative proof 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 B.10 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 proof. We have not talked about the case that $P_1P_2P_3$ are collinear, which requires one application of Theorem 10.4, but we hope by this stage it is clear how we achieved this, and how powerful our "squeeze" theorem is in allowing us to move between walls of a maze in a geometrically intuitive way.

10.4.3 Putting it all Together

We have packaged the theorem into a single declarative proof consisting of 119 steps. That we can manage such long proofs shows how easily Mizar Light, with our additional automated tools, can scale to long and complex proofs.

We would like to draw special attention to the hypotheses in the verified theorem (10.5). 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 $P_2P_3 \ldots P_n$, and that P_2P_3 does not intersect the rest of the path $P_3P_4 \ldots P_n$. These are the minimal hypotheses we need to apply Theorem 10.4 as described in this section. We can think of them as saying that the polygon appears *locally* simple.

$$
\begin{aligned}
&\vdash \texttt{between}\ P_1\ X'\ P_2 \land P_2 \neq P_3 \\
&\quad \land \neg \texttt{on\_polypath}\ (P_1\ :\ P_2\ :\ P_3\ :\ Ps)\ X \land \neg \texttt{on\_polypath}\ (P_3\ :\ Ps)\ P_2 \\
&\quad \land \neg(\exists Z.\texttt{between}\ X\ Z\ X' \land \texttt{on\_polypath}\ (P_1\ :\ P_2\ :\ P_3\ :\ Ps)\ Z) \\
&\quad \land \neg(\exists Z.\texttt{between}\ P_1\ Z\ P_2 \land \texttt{on\_polypath}\ (P_2\ :\ P_3\ :\ Ps)\ Z) \\
&\quad \land \neg(\exists Z.\texttt{between}\ P_2\ Z\ P_3 \land \texttt{on\_polypath}\ (P_3\ :\ Ps)\ Z) \\
&\quad \implies \exists Y\ Y'.\texttt{path\_connected}\ (\texttt{on\_polypath}\ (P_1\ :\ P_2\ :\ P_3\ :\ Ps))\ X\ Y \\
&\qquad \land \texttt{between}\ P_2\ Y'\ P_3 \land \neg \texttt{on\_polypath}\ (P_1\ :\ P_2\ :\ P_3\ :\ Ps)\ Y) \\
&\qquad \land \neg \exists Z.\texttt{between}\ Y\ Z\ Y' \land (P_1\ :\ P_2\ :\ P_3\ :\ Ps)\ Z)
\end{aligned}
\tag{10.5}
$$

10.5 Without-Loss-of-Generality

In the last section, we described how to move wall-to-wall in a polygon. However, we effectively made a without-loss-of-generality assumption when stating the theorem, since we assumed that the walls in question were defined by the first three elements of the vertex list. More generally, they could be the two walls defined by three adjacent elements of the list, or, if we are considering moving forward from the last wall of a polygon back to the first, then the vertices defining the walls come from the last two elements and the first two elements.

As with the sort of without-loss-of-generality assumptions considered by Harrison [26], we can justify ourselves if we can find an equivalence relation which preserves the properties of interest to us. We are, in effect, saying that before we move a player to the next wall, we must be able to transform the polygon's vertex list into the player's perspective, much as we would perform a coordinate transform.

10.5.1 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. This is not very geometric, but on the upside, the ideas in this section 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 halfs. Formally:

$$\texttt{rotation\_of} : [\alpha] \to [\alpha] \to \texttt{bool}$$

$$\texttt{rotation\_of}\; ps\; qs \iff \exists xs\; ys.\; ps = xs + +ys \wedge qs = ys + +xs.$$

This defines an equivalence relation[1], 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:

$$\begin{aligned}\texttt{rotation\_of}\; Ps\; Qs \iff \exists P\; Q\; Ps\; Qs.\; & Ps = ([P] + +Ps' + +[Q] + +Ps'' + +[P]) \\ & \wedge Qs = ([Q] + +Ps'' + +[P] + +Ps' + +[Q]) \\ & \vee Ps = Qs\end{aligned}$$

We verified that this is also an equivalence relation.

10.5.2 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 this late stage, when so much of the earlier verification was focused on synthetic

[1]Thanks to Steven Obua for helping me with the verification.

declarative geometry. But with perseverance, we obtained the necessary theorems:

$$\begin{aligned}\vdash & \texttt{rotation\_of}\; Ps\; Qs \\ & \Longrightarrow (\texttt{simple\_polygon}\; Ps \Longrightarrow \texttt{simple\_polygon}\; Qs) \\ & \quad \wedge \texttt{on\_polypath}\; Ps = \texttt{on\_polypath}\; Qs\end{aligned} \tag{10.6}$$

10.5.3 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 a wall of the polygon. We can formally describe the position of X' with:

$$\exists P_i\; P_{i+1}.\; \texttt{mem}\; (P_i, P_{i+1})\; (\texttt{adjacent}\; Ps) \wedge \texttt{between}\; P_i\; X'\; P_{i+1}.$$

The presence of `adjacent` is troublesome, but we have proved a theorem to help us out here:

$$\vdash \texttt{mem}\; (x,y)\; (\texttt{adjacent}\; xs) \iff \exists ys\; zs.\; xs = ws + +[x,y] + +zs$$

This gives us just about everything we need to perform a rotation. We know that if the witnessed ys is empty, then our wall $[x,y]$ must already be at the front of the list. Otherwise, we know that the list xs is of the form

$$xs = [w] + +ws + +[x,y] + +zs + +[w]$$

which is a rotation of

$$xs = [x,y] + +zs + +[w] + +ws + +[x]$$

This puts the wall at the front of the list, as we require to apply Theorem 10.5. Our invariance theorem (10.6) assures us that the rotation will not change the set of points defined by the vertex list, nor affect the simplicity of the polygon.

10.6 Moving to Any Wall

At this stage, we are able to move between any two walls of a polygon. This takes us a significant way towards a verification of this half of the Polygonal Jordan Curve

Theorem, and is a significant step towards verifying our alternative sketch proof from Chapter 7.

To formalise it, we say that, in particular, we have shown that, given a point X with line-of-sight to a point X' on some wall, there must be a path-connected point Y which has line-of-sight to a point Y' on the *first* wall. Formally, we verified:

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ Ps \land \neg\texttt{on\_polypath}\ Ps\ X \\
&\quad \land \texttt{mem}\ (P, Q)\ (\texttt{adjacent}\ Ps) \land \texttt{between}\ P\ X'\ Q \\
&\quad \neg(\exists Z. \texttt{between}\ X\ Z\ X' \land \texttt{on\_polypath}\ Ps\ Z) \\
&\qquad\qquad \implies \exists Y\ Y'.\ \texttt{path\_connected}\ (\texttt{on\_polypath}\ Ps)\ X\ Y \\
&\qquad\qquad\qquad\qquad \land \texttt{between}\ (\texttt{head}\ Ps)\ Y'\ (\texttt{head}\ (\texttt{tail}\ Ps)) \land \neg\texttt{on\_polypath}\ Ps\ Y \\
&\qquad\qquad\qquad\qquad \land \neg\exists Z. \texttt{between}\ Y\ Z\ Y' \land \texttt{on\_polypath}\ Ps\ Z
\end{aligned}
\tag{10.7}
$$

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 10.5, which we have said assumes that a polygon is only “locally” simple:

1. $\neg\texttt{on\_polypath}\ (P_3 : Ps)\ P_2$

2. $\neg(\exists Z. \texttt{between}\ P_1\ Z\ P_2 \land \texttt{on\_polypath}\ (P_2 : P_3 : Ps)\ Z)$

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 wall. The second will require that the walls 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 10.15, we have reproduced the example from §10.1, showing three players navigating a maze in order to have line-of-sight to the wall 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 10.4 using the auxiliary dashed lines. Each dashed line marks a point on a player's line-of-sight that they move towards, or a point on a wall towards which they rotate. Thus, we see each player navigating without compass or ruler, using only incidence and ordering.

...

`so consider` Z `such that between` $Y\ Z\ Y' \wedge$ `on_polypath` $[P_1, P_2, P_3]\ Z$

`from` ... `by` (8.1) 24

`obviously` (`by_ncols` $\circ$ `conjuncts`) `hence` $\neg$`on_polypath` $[P_2, P_3]\ Z$ `from` ...

`by` (8.1), (I, 1), (II, 1)

`hence` $\neg$`on_polypath` $[P_1, P_2]\ Z$ `from` 24 `by` (8.1) 25

`hence on_half_plane` $hp\ Y'$ `from` 12, 14, 23 `by` (B.10)

`hence on_half_plane` $hp\ Z$ `from` ..., 24 `by` (B.11) 26

`hence between` $P_1\ Z\ P_2$ `from` ..., 25 `by` (8.1), (B.7)

`qed from` ..., 26 `by` (I, 2), (II, 1), (B.7)

Figure 10.12: Verification Extract for the Convex Case of Theorem 10.5

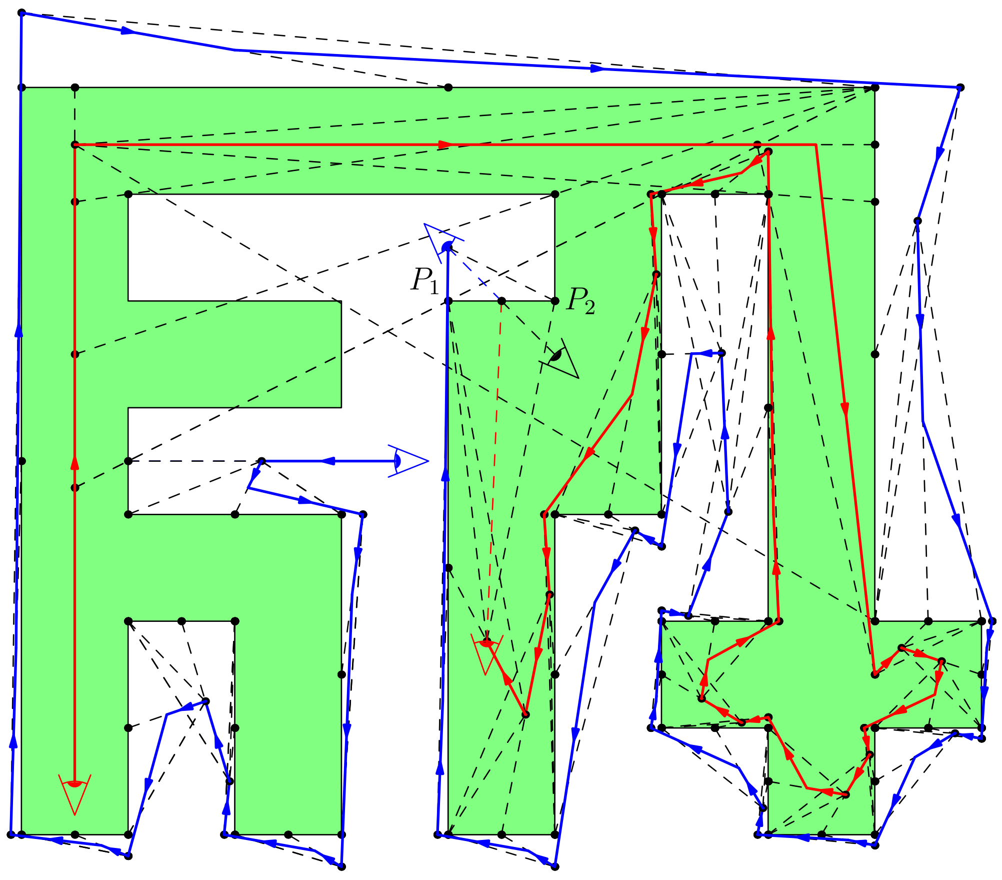


Figure 10.15: Navigating a Maze with Theorem 10.4

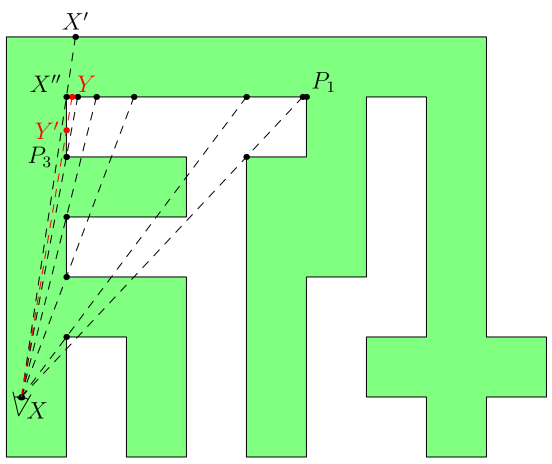


Figure 10.16: Obtaining Line-of-sight with a Wall

10.7 Final Steps

We are almost there. To finish the proof, we will need to show that when two points are on the same side of the same wall, then they are path-connected. We shall deal with this matter in §10.7.2. First, we shall deal with the neglected matter of how we are able to obtain line-of-sight to a wall in the first place. Both problems will have us reusing our near ubiquitous Theorem 10.4.

10.7.1 Getting onto the Maze

If we take an arbitrary point X in the plane and an arbitrary point X' on a simple polygon, 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 a wall.

However, if X'' coincides with a vertex as shown in Figure 10.16, we will need to 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 wall P_1P_2 or P_2P_3.

We can do this by applying Theorem 10.4 to the wall P_1P_2 and the polygonal fragment $P_3P_4 \dots P_n$ as described in §10.3.3.2. 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 wall P_1P_2, because the wall 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 reasoning tactic verifies that it is the required line-of-sight.

We have formalised the result retaining the without-loss-of-generality assumption. This allows the reader to review which hypotheses are actually needed for this particular theorem. This 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*:

$$\begin{aligned}
&\vdash \neg\texttt{between}\,P_1\,P_3\,P_2 \wedge \neg\texttt{between}\,P_2\,P_1\,P_3 \wedge P_1 \neq P_2 \wedge P_1 \neq P_3\\
&\quad \wedge \neg\texttt{on\_polypath}\,(P_3 : Ps)\,P_2\\
&\quad \wedge (\neg\exists Z.\texttt{between}\,P_1\,Z\,P_2 \wedge \texttt{on\_polypath}\,(P_2 : P_3 : Ps)\,Z)\\
&\quad \wedge (\neg\exists Z.\texttt{between}\,P_2\,Z\,P_3 \wedge \texttt{on\_polypath}\,(P_3 : Ps)\,Z)\\
&\quad \wedge \neg\texttt{on\_polyseg}\,(P_1 : P_2 : P_3 : Ps)\,X\\
&\quad \wedge (\neg\exists Z.\texttt{between}\,P_2\,Z\,X \wedge \texttt{on\_polypath}\,(P_1 : P_2 : P_3 : Ps)\,Z)\\
&\quad \Longrightarrow \exists Y.(\texttt{between}\,P_1\,Y\,P_2 \vee \texttt{between}\,P_2\,Y\,P_3)\\
&\qquad\quad \wedge \neg(\exists Z.\texttt{between}\,X\,Z\,Y \wedge \texttt{on\_polypath}\,(P_1 : P_2 : P_3 : Ps)\,Z)
\end{aligned} \tag{10.8}$$

We can now put any point on to the maze, and thus we can connect every point in the plane to another point with line-of-sight to any wall. This means that, if we have three points in the plane, we can find three paths which bring them together at the same wall. It is enough now to verify that two of the three points are segment-connected.

10.7.2 There are at most Two Regions

Here is where we are: we have three points with line-of-sight to a wall 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 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 most of the points obtained in our proofs up to now ultimately rely on Axiom II, 2. This axiom allows us to extend a line-segment, but there are many possible witnesses for its

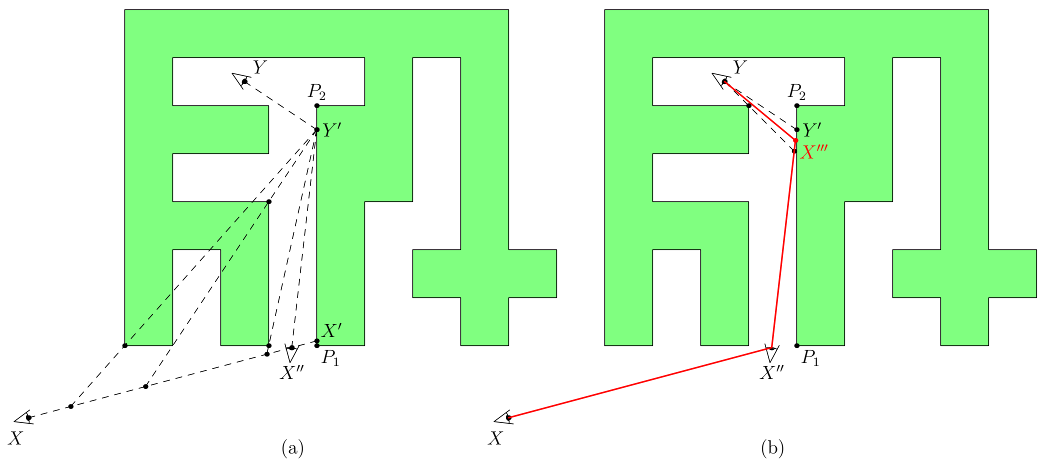


Figure 10.17: Connecting the Final Points

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 10.17(a). Here, our points X and Y have line-of-sight to different points on the same wall, 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 10.4 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 side of P_1P_2. This is needed because we only applied Theorem 10.4 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 10.18. We have omitted references to earlier steps.

$$\begin{aligned}
&\vdash \neg\texttt{on\_polypath}\ (P_1 : P_2 : Ps)\ X \wedge \neg\texttt{on\_polypath}\ (P_1 : P_2 : Ps)\ Y \\
&\quad \wedge \texttt{between}\ P_1\ X'\ P_2 \wedge \texttt{between}\ P_1\ Y'\ P_2 \\
&\quad \wedge \neg(\exists Z.\texttt{between}\ X\ Z\ X' \wedge \texttt{on\_polypath}\ (P_1 : P_2 : Ps)\ Z) \\
&\quad \wedge \neg(\exists Z.\texttt{between}\ Y\ Z\ Y' \wedge \texttt{on\_polypath}\ (P_1 : P_2 : Ps)\ Z) \\
&\quad \wedge \neg(\exists Z.\texttt{between}\ P_1\ Z\ P_2 \wedge \texttt{on\_polypath}\ (P_2 : Ps)\ Z) \\
&\quad \wedge \texttt{on\_line}\ P1\ \texttt{line\_of\_half\_plane}\ hp) \wedge \texttt{on\_line}\ P2\ \texttt{line\_of\_half\_plane}\ hp) \\
&\quad \wedge \texttt{on\_half\_plane}\ hp\ X \wedge \texttt{on\_half\_plane}\ hp\ Y \\
&\quad \implies \texttt{path\_connected}\ (\texttt{on\_polypath}\ (P_1 : P_2 : Ps)\ X\ Y
\end{aligned} \tag{10.9}$$

One final remark: in our sketch proof from §7.4, we actually assumed we could carry out this argument when establishing IH3 and IH5. Once again, we are making progress towards a verification of our first point-in-polygon proof. It is interesting to realise that, even if the two proofs are conceptually very different, many of the same arguments are needed for both.

10.8 Conclusion

The proof of the final theorem 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 a wall of the maze, and then use polygon rotations and Theorem 10.5 to find three paths to three points with line-of-sight to the first wall of the maze. We then apply Theorem 6.7 which says that two of these points must be on the same side of the first wall, and thus, from Theorem 10.9, we know that that these two points are path-connected.

As with the final theorem in the last chapter, the final theorem here is much more readable than the lemmas considered up to now. The theory so far builds theorems with large numbers of complicated hypotheses, such as those for Theorem 10.9 (and remember that we have not even bothered to mention all the planar hypotheses needed for every one of these theorems). When developing a theory such as this, where theorems are almost made opaque by their hypotheses, we have to pay particularly close attention to be confident that we are making progress. It was some relief when the development paid off.

```
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 $(B.10), (B.11)$
have $\forall Z.$on_half_plane $hp\ Z \implies \neg$on_polypath $[P_1, P_2]\ Z$ from $3, 8$
    by $(8.1), (I, 2), (II, 1), (B.7)$
hence $\forall Z.$on_polypath $[X'', X'', Y]\ Z \implies \neg$on_polypath $[P_1, P_2]\ Z$
    from... by $(8.1)$                                                    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 $(8.1), (II, 1)$
```

Figure 10.18: Verification Extract for Theorem 10.9

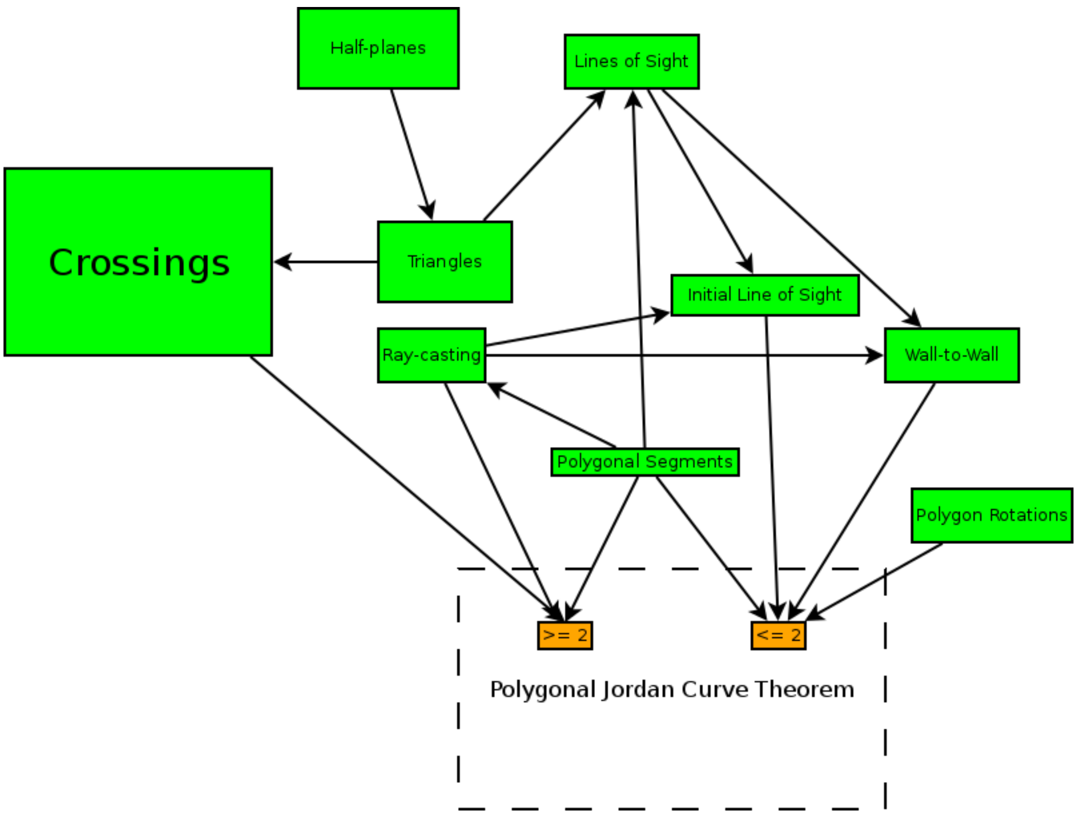


Figure 10.19: The Theory of Polygons

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ Ps \\
&\quad \wedge \neg \texttt{on\_polyseg}\ Ps\ P \wedge \neg \texttt{on\_polyseg}\ Ps\ Q \wedge \neg \texttt{on\_polyseg}\ Ps\ R \\
&\quad \Longrightarrow\ \texttt{seg\_connected}\ (\texttt{on\_polyseg}\ Ps)\ P\ Q \\
&\qquad\quad \vee \texttt{seg\_connected}\ (\texttt{on\_polyseg}\ Ps)\ P\ R \\
&\qquad\quad \vee \texttt{seg\_connected}\ (\texttt{on\_polyseg}\ Ps)\ Q\ R
\end{aligned}
$$

To end this chapter, we provide the diagram in Figure 10.19 showing the major dependencies needed to prove both parts of the Polygonal Jordan Curve Theorem. Individual boxes show the major hurdles in the theory, with their size roughly correlated with the size of the proofs needed to overcome that particular hurdle.

Of the 11000 lines of code needed for the whole project, our theory of polygons comes to a modest 5000 lines of mostly synthetic proof (there are far fewer actual proof *steps*). We do not believe this would have been achievable had we not had the support of our incidence discoverer, which justified 111 of our proof steps.

As Hilbert and Veblen suggested, the theory of half-planes is crucial to the proof of the Polygonal Jordan Curve Theorem, relieving us of the burden of applying Pasch's Axiom (II, 4). In fact, this axiom was needed on only 11 occasions. But besides the theory of half-planes, linear ordering also had pride of place. Our automated linear reasoning tactic was used an impressive 82 times.

We hope that our formal development contributes an interesting take on the Polygonal Jordan Curve Theorem. Our analysis has had to be far finer than that considered by either Veblen or Feigl, and thus we have introduced some new ideas to understand the mechanics of the proof. In particular, we would like to believe that the use of intuitive metaphors such as "lines-of-sight" in this chapter has kept things lively and geometrically focused, and made the details of our proofs all the more clear.

Most importantly, though, as is the case with all formal verification projects, we have forever set the validity of the mathematical arguments in these chapters beyond reproach. The question of whether the Polygonal Jordan Curve Theorem is a theorem of ordered geometry is now settled definitively even to the most *hardened* sceptic, who in this case, has taken the guise of an extremely pedantic computer system: HOL Light.

Bibliography

[1] Y. Balashov and M. Janssen. Presentism and relativity. *The British journal for the philosophy of science*, 54(2):327–346, 2003.

[2] Richard Boulton. *Efficiency in a Fully-Expansive Theorem Prover*. PhD thesis, Cambridge University, 1993.

[3] Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques Fleuriot, editors, *Automated Deduction in Geometry 2012*, Edinburgh, United Kingdom, September 2012. Jacques Fleuriot.

[4] B. Buchberger. A Theoretical Basis for the Reduction of Polynomials to Canonical Forms. *ACM SIGSAM Bull.*, 10(3):19–29, 1976.

[5] Ze-wei Chen. Efficient Access to Knowledge via Forward Chaining Tactics. Technical report, Cornell University, April 1995.

[6] Alonzo Church. A set of postulates for the foundation of logic. *Annals of Mathematics*, 33(2):pp. 346–366, 1932.

[7] Alonzo Church. A Formulation of the Simple Theory of Types. *Association for Symbolic Logic*, 5:56–68, 1940.

[8] Simon Colton, Alan Bundy, and Toby Walsh. On the notion of interestingness in automated mathematical discovery. *Int. J. Hum.-Comput. Stud.*, 53(3):351–375, 2000.

[9] Haskell B. Curry. The inconsistency of certain formal logic. *The Journal of Symbolic Logic*, 7(3):pp. 115–117, 1942.

[10] 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, POPL ’82, pages 207–212, New York, NY, USA, 1982. ACM.

[11] 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.

[12] 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.

[13] Radu Diaconescu. Axiom of choice and complementation. *Proceedings of the American Mathematical Society*, 51(1):pp. 176–178, 1975.

[14] 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.

[15] Matti Eklund. On how logic became first-order. *Nordic Journal of Philosophical Logic*, 1(2):147–67, 1996.

[16] Euclid. Elements. http://aleph0.clarku.edu/ djoyce/java/elements/elements.html, 1998.

[17] Solomon Feferman. Mathematical intuition vs. mathematical monsters. *Synthese*, 125(3):317–332, 2000.

[18] Georg Feigl. Über die elementaren Anordnungssätz. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 33, 1925.

[19] H. Guggenheimer. The Jordan Curve Theorem and an Unpublished Manuscript by Max Dehn. *Archive for History of Exact Sciences*, 17:193–200, 1977.

[20] Thomas C. Hales. Jordan’s Proof of the Jordan Curve Theorem. *Studies in Logic, Grammar and Rhetoric*, 10(23), 2007.

[21] Thomas C. Hales. The Jordan Curve Theorem, Formally and Informally. *The American Mathematical Monthly*, 114:882894, 2007.

[22] John Harrison. Formalized mathematics. Technical Report 36, Turku Centre for Computer Science (TUCS), Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland, 1996. Available on the Web as `http://www.cl.cam.ac.uk/~jrh13/papers/form-math3.html`.

[23] 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.

[24] 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.

[25] John Harrison. The hol light manual (1.0). `http://www.cl.cam.ac.uk/~jrh13/hol-light/manual-1.1.pdf`, 1998.

[26] 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.

[27] Henri Poincaré. Poincaré's Review of Hilbert's "Foundations of Geometry". *Bulletin of the American Mathematical Society*, 10(1):1–23, 1903.

[28] Henri Poincaré. *Science and Method.* Cosimo Classics, 2007.

[29] David Hilbert. On the Infinite. In Paul Benacerraf and Hilary Putnam, editors, *Philosophy of Mathematics: Selected Readings*. Cambridge University Press, 1984.

[30] Peter V. Homeier. The hol-omega logic. In *TPHOLs*, pages 244–259, 2009.

[31] 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.

[32] 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.

[33] 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.

[34] Jordan, Camille. *Cours d'analyse de l'École polytechnique*. Gauthier-Villars et fils Paris, 1893.

[35] 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.

[36] Vladimir Kanovei and Michael Reeken. A nonstandard proof of the Jordan curve theorem. *Pacific Journal of Mathematics*, 36(1):219–229, 1971.

[37] H. C. Kennedy. Origins of Modern Axiomatics: Pasch to Peano. *The American Mathematical Monthly*, 79:133–136, 1972.

[38] Oleg Kiselyov, Chung chieh Shan, Daniel P. Friedman, and Amr Sabry. Backtracking, interleaving, and terminating monad transformers: (functional pearl). In *ICFP*, pages 192–203, 2005.

[39] Wilbur Richard Knorr. *The Evolution of the Euclidean Elements*. Springer, February 1974.

[40] Ramana Kumar and Tjark Weber. Validating qbf validity in hol4. In *ITP*, pages 168–183, 2011.

[41] 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.

[42] N. J. Lennes. Theorems on the Simple Finite Polygon and Polyhedron. *American Journal of Mathematics*, 33(1):pp. 37–62, 1911.

[43] Nicolas Magaud, Julien Narboux, and Pascal Schreck. Formalizing Desargues' theorem in Coq using ranks. In *Symposium on Applied Computing*, pages 1110–1115, 2009.

[44] Conor McBride and Ross Paterson. Applicative programming with effects. *Journal of Functional Programming*, 18(1):1–13, 2008.

[45] Roy L. McCasland and Alan Bundy. MATHsAiD: A Mathematical Theorem Discovery Tool. In *SYNASC*, pages 17–22, 2006.

[46] Robert McNaughton. Review: Alfred Tarski, A Decision Method for Elementary Algebra and Geometry. *Bulletin of the American Mathematical Society*, 59(1):91–93, 1953.

[47] Michael A. McRobbie and John K. Slaney, editors. *Optimizing proof search in model elimination*, volume 1104 of *Lecture Notes in Computer Science*. Springer, 1996.

[48] 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.

[49] G. H. Meisters. Polygons have ears. *The American Mathematical Monthly*, 82(6):pp. 648–651, 1975.

[50] Elliot Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall, 4th edition, 1997.

[51] Jia Meng, Claire Quigley, and Lawrence C. Paulson. Automation for interactive proof: First prototype. *Inf. Comput.*, 204(10):1575–1596, 2006.

[52] Christopher P Wadsworth Michael J Gordon, Arthur J Milner. *Edinburgh LCF*. Springer-Verlag, 1979.

[53] 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.

[54] Robin Milner, Mads Tofte, Robert Harper, and David Macqueen. *The Definition of Standard ML - Revised*. The MIT Press, rev sub edition, May 1997.

[55] 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.

[56] Alan Musgrave. Logicism revisited. *The British Journal for the Philosophy of Science*, 28(2):pp. 99–127, 1977.

[57] Julien Narboux. Mechanical Theorem Proving in Tarski's Geometry. In *Automated Deduction in Geometry*, volume 4869, pages 139–156, 2006.

[58] Victor Pambuccian. Forms of the Pasch axiom in ordered geometry. *Mathematical Logic Quarterly*, 56(1):29–34, 2010.

[59] Victor Pambuccian. The axiomatics of ordered geometry: I. ordered incidence spaces. *Expositiones Mathematicae*, 29(1):24 – 66, 2011.

[60] Lawrence C. Paulson. *Isabelle: a Generic Theorem Prover*. Number 828 in Lecture Notes in Computer Science. Springer – Berlin, 1994.

[61] Art Quaife. Automated Development of Tarski's Geometry. *J. Autom. Reasoning*, 5(1):97–118, 1989.

[62] Richard Courant and Herbert Robbins. *"What is Mathematics?"*. Oxford University Press, 1996.

[63] Bertrand Russell. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30(3):pp. 222–262, 1908.

[64] Jeremy Yallop Sam Lindley and Phil Wadler. Idioms are oblivious, arrows are meticulous, monads are promiscuous. In *Mathematically Structured Functional Programming*, 2008. no proceedings.

[65] Phil Scott. Mechanising Hilbert's *Foundations of Geometry* in Isabelle. Master's thesis, University of Edinburgh, 2008.

[66] 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.

[67] 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.

[68] Shang-Ching Chou. *Mechanical Geometry Theorem Proving*. D. Reidel Publishing Company, 1988.

[69] Shang-Ching Chou, Xiao-Shan Gao, Jing-Zhong Zhang. *Machine Proofs in Geometry*. World Scientific Publishing, 1994.

[70] J. Michael Spivey. Algebras for combinatorial search. *Journal of Functional Programming*, 19(3-4):469–487, 2009.

[71] Sana Stojanovic, Vesna Pavlovic, and Predrag Janicic. A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs. In *Automated Deduction in Geometry*, pages 201–220, 2010.

[72] 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.

[73] Alfred Tarski and Steven Givant. Tarski's System of Geometry. *Bull. Symbolic Logic*, 5:175–214, 1999.

[74] Wen tsün Wu. *Mechanical Theorem Proving in Geometries.* Springer-Verlag Wien New York, 1994.

[75] Helge Tverberg. A Proof of the Jordan Curve Theorem. *Bulletin of the London Mathematical Society*, 12:34–38, 1980.

[76] Oswald Veblen. Hilbert's Foundations of Geometry. *The Monist*, 13(2):pp. 303–309, 1903.

[77] Oswald Veblen. A System of Axioms for Geometry. *Transactions of the American Mathematical Society*, 5:343–384, 1904.

[78] Oswald Veblen. Theory on Plane Curves in Non-Metrical Analysis Situs. *Transactions of the American Mathematical Society*, 6(1):83–98, 1905.

[79] Philip Wadler. Monads for functional programming. In *Advanced Functional Programming*, pages 24–52, 1995.

[80] Hermann Weyl. David Hilbert and his mathematical work. *Bulletin of the American Mathematical Society*, 50:635, 1944.

[81] A.N. Whitehead and B. Russell. *Principia mathematica.* Number v. 2 in Principia Mathematica. University Press, 1912.

[82] Freek Wiedijk. Mizar Light for HOL Light. In *Theorem Proving in Higher Order Logics*, volume 2152, pages 378–394, London, UK, 2001. Springer-Verlag.

[83] Freek Wiedijk, editor. *The Seventeen Provers of the World, Foreword by Dana S. Scott*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.

[84] Freek Wiedijk. Mizar's Soft Type System. In *Theorem Proving in Higher Order Logics*, volume 4732, pages 383–399. Springer, 2007.

[85] Pierre Castéran Yves Bertot. *Interactive Theorem Proving and Program Development*. Springer, 2004.

Appendix A

Group I Axioms and Elementary Consequences

A.1 Axioms

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

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

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

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

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

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

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

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

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

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

A.2 Elementary Consequences

$$\begin{aligned}&\exists A\ B\ C.\texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha\\ &\qquad\wedge \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\end{aligned} \tag{2.2.4}$$

Appendix B

Group II Axioms and Elementary Consequences

B.1 Axioms

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

$$A \neq B \implies \exists C. \texttt{between}\,A\,B\,C \tag{II, 2}$$

$$\texttt{between}\,A\,B\,C \implies \neg\texttt{between}\,A\,C\,B \tag{II, 3}$$

Pasch's Axiom

$$\begin{aligned} & \neg(\exists a. \texttt{on\_line}\,A\,a \land \texttt{on\_line}\,B\,a \land \texttt{on\_line}\,C\,a) \\ & \land \texttt{on\_plane}\,A\,\alpha \land \texttt{on\_plane}\,B\,\alpha \land \texttt{on\_plane}\,C\,\alpha \\ & \land (\forall P. \texttt{on\_line}\,P\,a \implies \texttt{on\_plane}\,P\,\alpha) \\ & \neg\texttt{on\_line}\,A\,a \land \neg\texttt{on\_line}\,B\,a \land \neg\texttt{on\_line}\,C\,a \\ & \land \texttt{on\_line}\,P\,a \land \texttt{between}\,A\,P\,B \\ & \implies \exists Q. \texttt{on\_line}\,Q\,a \land (\texttt{between}\,A\,Q\,C \lor \texttt{between}\,B\,Q\,C) \end{aligned} \tag{II, 4}$$

B.2 Elementary Consequences

$$\begin{aligned}
&\neg\texttt{collinear}\ \{A,B,C\} \wedge \neg\texttt{collinear}\ \{A,D,E\} \wedge \neg\texttt{collinear}\ \{C,D,E\} && (2.13)\\
&\wedge \texttt{planar}\ \{A,B,C,D,E\} \wedge \texttt{between}\ A\ D\ B && (B.1)\\
&\implies \exists F.\texttt{collinear}\ \{D,E,F\} \wedge (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C). && (B.2)
\end{aligned}$$

$$\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a)\\
&\wedge \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a)\\
&\wedge \neg(\exists a.\texttt{on\_line}\ C\ a \wedge \texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a)\\
&\wedge (\exists \alpha.\texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha\\
&\qquad \wedge \texttt{on\_plane}\ D\ \alpha \wedge \texttt{on\_plane}\ E\ \alpha) \qquad\qquad (B.3)\\
&\wedge \texttt{between}\ A\ D\ B\\
&\implies \exists F.(\exists a.\texttt{on\_line}\ D\ a \wedge \texttt{on\_line}\ E\ a \wedge \texttt{on\_line}\ F\ a)\\
&\qquad\qquad (\texttt{between}\ A\ F\ C \vee \texttt{between}\ B\ F\ C)
\end{aligned}$$

$$\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ C\\
&\qquad \implies \exists F.\texttt{between}\ D\ E\ F \wedge \texttt{between}\ A\ F\ B
\end{aligned} \tag{4.3}$$

$$\begin{aligned}
&\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \wedge \texttt{between}\ B\ C\ D \wedge \texttt{between}\ A\ E\ B\\
&\qquad \implies \exists F.\texttt{between}\ D\ F\ E \wedge \texttt{between}\ A\ F\ C.
\end{aligned} \tag{4.4}$$

$$A \neq C \to \exists D.\texttt{between}\ A\ D\ C \tag{Theorem 3}$$

$$\begin{aligned}
&\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a\\
&\wedge\ A \neq B \wedge A \neq C \wedge B \neq C\\
&\implies \texttt{between}\ A\ B\ C \vee \texttt{between}\ B\ A\ C \vee \texttt{between}\ A\ C\ B
\end{aligned} \tag{Theorem 4}$$

$$\begin{aligned}
&(\texttt{between}\ A\ B\ C \wedge \texttt{between}\ B\ C\ D \implies \texttt{between}\ A\ B\ D \wedge \texttt{between}\ A\ C\ D)\\
&\wedge (\texttt{between}\ A\ B\ C \wedge \texttt{between}\ A\ C\ D \implies \texttt{between}\ A\ B\ D \wedge \texttt{between}\ B\ C\ D)
\end{aligned} \tag{Theorem 5}$$

$$A \neq B \implies \mathit{INFINITE}\ \{P | \mathtt{between}\ A\ P\ B\} \tag{5.17}$$

$$\begin{aligned}&\neg\mathtt{on\_line}\ A\ a \wedge \neg\mathtt{on\_line}\ B\ a \wedge \neg\mathtt{on\_line}\ C\ a\\ &\wedge \mathtt{on\_line}\ D\ a \wedge \mathtt{on\_line}\ E\ a \wedge \mathtt{on\_line}\ F\ a\\ &\implies \neg\mathtt{between}\ A\ D\ B \vee \neg\mathtt{between}\ A\ E\ C \vee \neg\mathtt{between}\ B\ F\ C\end{aligned} \tag{6.6}$$

B.3 Half-Planes

$$\exists P.\mathtt{on\_half\_plane}\ hp\ P \tag{B.4}$$

$$\begin{aligned}&\mathtt{on\_half\_plane}\ hp\ P \wedge \mathtt{on\_half\_plane}\ hq\ P\\ &\wedge \mathtt{line\_of\_half\_plane}\ hp = \mathtt{line\_of\_half\_plane}\ hq\\ &\implies hp = hq\end{aligned} \tag{B.5}$$

$$\begin{aligned}&\mathtt{on\_half\_plane}\ hp\ P \wedge \mathtt{on\_plane}\ P\ \alpha\\ &\wedge (\forall R.\ \mathtt{on\_line}\ R\ (\mathtt{line\_of\_half\_plane}\ hp) \implies \mathtt{on\_plane}\ R\ \alpha)\\ &\implies \mathtt{on\_half\_plane}\ hp\ Q \implies \mathtt{on\_plane}\ Q\ \alpha\end{aligned} \tag{B.6}$$

$$\mathtt{on\_half\_plane}\ hp\ P \implies \neg\mathtt{on\_line}\ P\ (\mathtt{line\_of\_half\_plane}\ hp) \tag{B.7}$$

$$\begin{aligned}&\neg(\exists a.\mathtt{on\_line}\ A\ a \wedge \mathtt{on\_line}\ B\ a \wedge \mathtt{on\_line}\ C\ a)\\ &\implies \exists! hp.\mathtt{on\_line}\ A\ (\mathtt{line\_of\_half\_plane}\ hp)\\ &\qquad\qquad \wedge \mathtt{on\_line}\ B\ (\mathtt{line\_of\_half\_plane}\ hp)\\ &\qquad\qquad \wedge \mathtt{on\_half\_plane}\ hp\ C.\end{aligned} \tag{B.8}$$

$$\begin{aligned}&(\forall P.\mathtt{on\_line}\ P\ a \implies \mathtt{on\_plane}\ P\ \alpha)\\ &\implies \exists hp\ hq.\ hp \neq hq\\ &\quad \wedge a = \mathtt{line\_of\_half\_plane}\ hp \wedge a = \mathtt{line\_of\_half\_plane}\ hq\\ &\quad \wedge (\forall P.\mathtt{on\_plane}\ P\ \alpha\\ &\qquad \iff \mathtt{on\_line}\ P\ a \vee \mathtt{on\_half\_plane}\ hp\ P \vee \mathtt{on\_half\_plane}\ hq\ P)\end{aligned} \tag{6.7}$$

$$\begin{aligned}&(\forall R.\ \texttt{on\_half\_plane}\ hp\ R \implies \texttt{on\_plane}\ R\ \alpha) \land \texttt{on\_half\_plane}\ hp\ P\\ &\implies (\texttt{on\_half\_plane}\ hp\ Q\\ &\qquad \iff \neg(\exists R.\texttt{on\_line}\ R\ (\texttt{line\_of\_half\_plane}\ hp) \land \texttt{between}\ P\ R\ Q)\\ &\qquad\qquad \land \texttt{on\_plane}\ Q\ \alpha \land \neg\texttt{on\_line}\ Q\ (\texttt{line\_of\_half\_plane hp}))\end{aligned} \tag{B.9}$$

$$\begin{aligned}&\texttt{on\_line}\ P\ (\texttt{line\_of\_half\_plane}\ hp) \land \texttt{on\_half\_plane}\ hp\ Q\\ &\implies \texttt{between}\ P\ Q\ R \lor \texttt{between}\ P\ R\ Q \implies \texttt{on\_half\_plane}\ hp\end{aligned} \tag{B.10}$$

$$\begin{aligned}&\texttt{on\_half\_plane}\ hp\ P \land \texttt{on\_half\_plane}\ hp\ R\\ &\implies \texttt{between}\ P\ Q\ R \implies \texttt{on\_half\_plane}\ hp\ Q\end{aligned} \tag{B.11}$$

B.4 Rays

$$\exists P.\texttt{on\_ray}\ r\ P \tag{B.12}$$

$$\texttt{on\_ray}\ r\ P \land \texttt{on\_ray}\ s\ P \land \texttt{ray\_origin}\ r = \texttt{ray\_origin}\ s \implies r = s \tag{B.13}$$

$$\begin{aligned}&\texttt{on\_ray}\ r\ P \land \texttt{on\_line}\ P\ a \land \texttt{on\_line}\ (\texttt{ray\_origin}\ r)\ a\\ &\implies \texttt{on\_ray}\ r\ Q \implies \texttt{on\_line}\ Q\ a\end{aligned} \tag{B.14}$$

$$\neg\texttt{on\_ray}\ r\ (\texttt{ray\_origin}\ r) \tag{B.15}$$

$$\begin{aligned}&\texttt{on\_line}\ P\ a\\ &\implies \exists r\ s.r \neq s \land P = \texttt{ray\_origin}\ r \land Q = \texttt{ray\_origin}\ s\\ &\qquad \land (\forall X.\texttt{on\_line}\ X\ a \iff X = \texttt{ray\_origin}\ r \lor \texttt{on\_ray}\ r\ X \lor \texttt{on\_ray}\ s\ X)\end{aligned} \tag{B.16}$$

$$\begin{aligned}&(\forall R.\texttt{on\_ray}\ r\ R \implies \texttt{on\_line}\ R\ a) \land \texttt{on\_ray}\ r\ P\\ &\implies (\texttt{on\_ray}\ r\ Q\\ &\qquad \iff Q \neq \texttt{ray\_origin}\ r \land \neg\texttt{between}\ P\ (\texttt{ray\_origin}\ r)\ Q \land \texttt{on\_line}\ Q\ a\end{aligned} \tag{B.17}$$

Appendix C

Polygonal Jordan Curve Theorem: Verified Theorems and Dependencies

C.1 HOL Light List and Set Library

Function specifications marked with † have been contributed.

$\texttt{head} : [\alpha] \to \alpha$	$\texttt{tail} : [\alpha] \to [\alpha]$†	$\texttt{length}\ [\alpha] \to \mathbb{N}$
$\texttt{head}\ (x : xs) = x$	$\texttt{tail}\ [] = []$	$\texttt{length}\ [] = 0$
	$\texttt{tail}\ (x : xs) = xs$	$\texttt{length}\ (x : xs) = \texttt{length}\ xs + 1$

$$\texttt{butlast} : [\alpha] \to [\alpha]$$
$$\texttt{butlast}\ [] = []$$
$$\texttt{butlast}\ (x : xs) = \texttt{if}\ xs = []\ \texttt{then}\ []\texttt{else}\ x : (\texttt{butlast}\ xs)$$

$\texttt{mem} : \alpha \to [\alpha] \to \texttt{bool}$	$\texttt{all} : (\alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool}$
$\texttt{mem}\ x\ [] = \bot$	$\texttt{all}\ p\ [] = \bot$
$\texttt{mem}\ x\ (y : ys) = x = y \vee \texttt{mem}\ x\ ys$	$\texttt{all}\ p\ (x : xs) = p\ x \wedge \texttt{all}\ p\ xs$

$$\texttt{pairwise} : (\alpha \to \alpha \to \texttt{bool}) \to [\alpha] \to \texttt{bool}$$
$$\texttt{pairwise}\ R\ [] = \top$$
$$\texttt{pairwise}\ R\ (x : xs) = \texttt{all}\ (Rx)\ xs \wedge \texttt{pairwise}\ R\ xs$$

$\texttt{zip} : [\alpha] \rightarrow [\beta] \rightarrow [(\alpha,\beta)]$

$\texttt{zip}\,[]\,[] = []$

$\texttt{zip}\,(x : xs)(y : ys) = (x,y) : \texttt{zip}\,xs\,ys$

$\texttt{adjacent} : [\alpha] \rightarrow [(\alpha,\alpha)]\dagger$

$\texttt{adjacent}\,xs = \texttt{zip}\,(\texttt{butlast}\,xs)\,(\texttt{tail}\,xs)$

$$\texttt{disjoint} : (\alpha \rightarrow \texttt{bool}) \rightarrow (\alpha \rightarrow \texttt{bool}) \rightarrow \texttt{bool}$$

$$\texttt{disjoint}\,S\,T = S \cap T = \emptyset$$

C.2 Polygon Definitions

$$\texttt{on\_polyseg} : [\texttt{point}] \rightarrow \texttt{point} \rightarrow \texttt{bool}$$

$$\texttt{on\_polyseg} Ps P \iff$$

$$\texttt{mem}\,P\,Ps \vee \exists x\,y.\,\texttt{mem}\,(x,y)\,(\texttt{adjacent}\,Ps) \wedge \texttt{between}\,x\,P\,y.$$

$$\texttt{seg\_connected} : \texttt{plane} \rightarrow (\texttt{point} \rightarrow \texttt{bool}) \rightarrow \texttt{point} \rightarrow \texttt{point}$$

$$\texttt{seg\_connected}\,\alpha\,\mathit{figure}\,P\,Q \iff$$

$$\exists \mathit{path}.\,\mathit{path} \neq []$$

$$\wedge\,(\forall R.\,\texttt{mem}\,R\,\mathit{path} \implies \texttt{on\_plane}\,R\,\alpha)$$

$$\wedge\,\texttt{head}\,\mathit{path} = P \wedge \texttt{last}\,\mathit{path} = Q$$

$$\wedge\,\texttt{disjoint}\,(\texttt{on\_polyseg}\,\mathit{path})\,\mathit{figure}.$$

$$\texttt{simple\_polygon} : [\texttt{point}] \rightarrow \texttt{bool}$$

$$\texttt{simple\_polygon}\,Ps \iff$$

$$3 \leq \texttt{length}\,Ps$$

$$\wedge\,\texttt{head}\,ps = \texttt{last}\,Ps$$

$$\wedge\,(\forall P.\,\texttt{mem}\,P\,Ps \implies \texttt{on\_plane}\,P\,\alpha)$$

$$\wedge\,\texttt{pairwise}\,(\neq)\,(\texttt{butlast}\,Ps)$$

$$\wedge\,\neg(\exists P\,Q\,X.\,\texttt{mem}\,X\,Ps \wedge \texttt{mem}\,(P,Q)\,(\texttt{adjacent}\,Ps) \wedge \texttt{between}\,P\,X\,Q)$$

$$\wedge\,\texttt{pairwise}\,(\lambda(P,Q)\,(P',Q').\,\neg(\exists X.\texttt{between}\,P\,X\,P' \wedge \texttt{between}\,Q\,X\,Q').$$

C.3 Theorems

$$
\begin{aligned}
&\vdash \texttt{on\_plane}\ P1\ \alpha \land \texttt{on\_plane}\ P2\ \alpha \land \texttt{on\_plane}\ Q1\ \alpha \land \texttt{on\_plane}\ Q2\ \alpha\\
&\quad \land (\forall X.\texttt{mem}\ X\ Ps \implies \texttt{on\_plane}\ X\ \alpha) \land (\forall X.\texttt{mem}\ X\ Qs \implies \texttt{on\_plane}\ X\ \alpha)\\
&\quad \land \neg(\exists a.\texttt{on\_line}\ P1\ a \land \texttt{on\_line}\ P2\ a \land \texttt{on\_line}\ Q1\ a)\\
&\quad \land \neg(\exists a.\texttt{on\_line}\ Q1\ a \land \texttt{on\_line}\ Q2\ a \land \texttt{on\_line}\ P1\ a)\\
&\quad \land \texttt{between}\ P1\ X\ P2 \land \texttt{between}\ Q1\ X\ Q2\\
&\quad \land P1 = \texttt{last}\ (P2 : Ps) \land Q1 = \texttt{last}\ (Q2 : Qs)\\
&\quad \implies \exists Y.\texttt{on\_polyseg}\ (P2 : Ps\ Y) \land \texttt{on\_polyseg}\ (Q1 : Q2 : Qs)\ Y\\
&\qquad \lor \texttt{on\_polyseg}\ (P1 : P2 : Ps)\ Y) \land \texttt{on\_polyseg}\ (Q2 : Qs)\ Y
\end{aligned}
$$

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ \alpha\ Ps\\
&\quad \implies \exists P\ Q.\ \texttt{on\_plane}\ P\ \alpha \land \texttt{on\_plane}\ Q\ \alpha\\
&\qquad \land \neg\texttt{on\_polyseg}\ Ps\ P \land \neg\texttt{on\_polyseg}\ Ps\ Q\\
&\qquad \land \neg\texttt{seg\_connected}\ \alpha\ (\texttt{on\_polyseg}\ Ps)\ P\ Q
\end{aligned}
$$

$$
\begin{aligned}
&\vdash \texttt{simple\_polygon}\ \alpha\ Ps\\
&\quad \land \texttt{on\_plane}\ P\ \alpha \land \texttt{on\_plane}\ Q\ \alpha \land \texttt{on\_plane}\ R\ \alpha\\
&\quad \land \neg\texttt{on\_polyseg}\ Ps\ P \land \neg\texttt{on\_polyseg}\ Ps\ Q \land \neg\texttt{on\_polyseg}\ Ps\ R\\
&\quad \implies \texttt{seg\_connected}\ \alpha\ (\texttt{on\_polyseg}\ Ps)\ P\ Q\\
&\qquad \lor \texttt{seg\_connected}\ \alpha\ (\texttt{on\_polyseg}\ Ps)\ P\ R\\
&\qquad \lor \texttt{seg\_connected}\ \alpha\ (\texttt{on\_polyseg}\ Ps)\ Q\ R
\end{aligned}
$$

Appendix D

Polygonal Jordan Curve Theorem: Supporting Theorems

$$\texttt{in\_triangle}\ (A,B,C)\ P \implies \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \tag{D.1}$$

$$\begin{aligned}
&Qs = [P] + +Ps + +[P] \\
&\wedge\, \texttt{on\_plane}\ A\ \alpha \wedge \texttt{on\_plane}\ B\ \alpha \wedge \texttt{on\_plane}\ C\ \alpha \wedge \texttt{on\_plane}\ C'\ \alpha \\
&(\forall X.\texttt{mem}\ X\ Qs \implies \texttt{on\_plane}\ X\ \alpha) \\
&\neg\texttt{on\_polyseg} Qs A \\
&\wedge\, (\texttt{between}\ A\ B\ B' \vee \texttt{between}\ A\ B'\ B \vee A \neq B \wedge B \neq B') \\
&\wedge\, \neg(\exists.\texttt{on\_polyseg}\ [B,B']\ X \wedge \texttt{on\_polyseg}\ Qs\ X) \\
&\implies \forall C\, C'.\neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B\ a \wedge \texttt{on\_line}\ C\ a) \\
&\qquad\qquad \wedge\, \neg(\exists a.\texttt{on\_line}\ A\ a \wedge \texttt{on\_line}\ B'\ a \wedge \texttt{on\_line}\ C'\ a) \\
&\qquad \implies \exists \Gamma'.\texttt{polyseg\_crossings}\ (A,B,C)\ (\Gamma_{final}\ (A,B,C)\ \Gamma\ (\texttt{adjacent}\ Qs)) \\
&\qquad\qquad\qquad\qquad (\texttt{adjacent}\ Qs) \\
&\qquad\qquad = \texttt{polyseg\_crossings}\ (A,B',C')\ (\Gamma_{final}\ (A,B',C')\ \Gamma'\ (\texttt{adjacent}\ Qs)) \\
&\qquad\qquad\qquad\qquad (\texttt{adjacent}\ Qs)
\end{aligned} \tag{9.29}$$

Appendix E

Translations

Throughout the thesis, we have tried to use conventional notation from mathematics (including type theory). This is not quite the syntax used by HOL Light, which lacks unicode support to render the symbols correctly. We also use various pieces of sugar to keep things more elegant.

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