EPiC Series in Computing
Volume 39, 2016, Pages 143–155
SCSS 2016. 7th International Symposium on Symbolic Computation in Software Science

Compass-free Navigation of Mazes

Phil Scott, Jacques Fleuriot[*]

University of Edinburgh

Abstract

If you find yourself in a corridor of a standard maze, a sure and easy way to escape is to simply pick the left (or right) wall, and then follow it along its twists and turns and around the dead-ends till you eventually arrive at the exit. But what happens when you cannot tell left from right? What if you cannot tell North from South? What if you cannot judge distances, and have no idea what it means to follow a wall in a given direction? The possibility of escape in these circumstances is suggested in the statement of an unproven theorem given in David Hilbert's celebrated *Foundations of Geometry*, in which he effectively claimed that a standard maze could be fully navigated using axioms and concepts based *solely* on the relations of points lying on lines in a specified order. We discuss our algorithm for this surprisingly challenging version of the maze navigation problem, and our HOL Light verification of its correctness from Hilbert's axioms.

1 Background

> Every mathematician knows how to walk without running into walls. Detailed figures [. . .] would be superfluous, if not downright insulting.
>
> Yet, it is quite another matter altogether to train a computer to navigate its way around a maze-like polygon without collisions.

So Tom Hales tells us [8], as he discusses his formal verification of the Jordan Curve Theorem in HOL Light [9]. Hales has some interesting commentary [7] on this verification, including a convincing vindication of Jordan's 1887 proof [13], ignored by folklore in favour of the purportedly more rigorous proof by Oswald Veblen [24]. Veblen presents *highly* detailed figures and lengthy arguments for how to walk simple polygonal arcs. Was he being grossly pedantic?

It turns out that Veblen was effectively working from two simple groups of axioms for geometry as presented in David Hilbert's *Foundations of Geometry* [11], a text well-known to the theorem proving community [1, 3, 18, 20, 21, 22]. Hilbert had stated the Jordan Curve Theorem for polygons but gave no proof, claiming one could be obtained "without serious difficulty". Veblen knew better, and Hilbert's claim of ease was deleted as a "correction" in later editions.

The fact is that Hilbert's axioms are so weak that they place a terrible handicap on the geometer, who must walk with scant defensives against running into the walls. They have no

[*]This research is supported by the University of Edinburgh ESPRC project (EP/L011794/1).

recourse to a theory of angle comparison from which to take a bearing. They have no knowledge of the parallel lines that would allow them to walk in the direction of a maze wall. There is no continuity of space nor notion of distance which would allow them to carefully approach without bumping their heads. Indeed, independence results exhibit geometries in which the axioms hold, but where space is non-Archimedean, meaning that mazes can have corridors of infinitesimally narrow width to squeeze through. And yet, for all his efforts, Veblen's proof is *still* deemed inconclusive [14], with the paucity of subsequent corrections having been described as "astonishing" [6].

Hales' verification goes through in the more restricted context of Euclidean space, where we do not suffer these handicaps. But even there, he acknowledged the challenge of teaching a computer to navigate his standard mazes, and he was simplifying things for himself by considering only those whose walls run along a grid. We shall, instead, be making things significantly harder for ourselves as we describe how to navigate Hilbert's polygons. Here, detailed diagrams are a must!

2 Related work

The MIZAR [2] community proposed and began working on a verification of the full Jordan Curve Theorem in 1991 for Euclidean geometry, and completed the special case for standard mazes in 1996. The rest of the verification was completed in 2005, the same year that Hales completed his own. The special cases took a restricted form. For MIZAR, only standard mazes with horizontal and vertical walls were considered. For Hales, the walls had to meet at the points on a grid. These verifications are not much use to us, since the propositions being verified here use concepts which are not even definable in our weak setting.

In 2009, Dufourd formally verified a constructive proof of the Discrete Jordan Curve Theorem [4], based on hypermaps. The verification is of particular value for computational topology, but to be of use to us, we would need to find a general way to embed hypermaps and their operations into our geometric setting.

As for prose proofs based on Hilbert's axioms, there are only a few. A proof by Feigl [5] is cited in a supplement to later editions of the *Foundations of Geometry*. Lennes [15], who criticised parts of Veblen's proof as "inconclusive", provides another proof. A much later proof was put forward by Main [16], and a few years after that, Guggenheimer [6] wrote a very elegant proof exploiting a theorem by Dehn, in which he was able to reduce the problem from arbitrary standard mazes to the trivial case of a triangle.

Our own verification had its genesis in an attempt to recursively decompose the problem via polygon triangulations [12], but we changed tack and decided to work directly from Veblen's proof, patching the holes that left Lennes and Guggenheimer sceptical as to its validity.

3 Axioms

The starting points for our problem are Hilbert's axioms for a bare "ordered geometry" or for "ordered incidence spaces", a topic which has seen some recent investigation [19]. If Euclidean geometry is classically the geometry of straight-edge/compass constructions, then ordered geometry is merely the geometry of the straight edge. The axioms provide us with a very simple vocabulary: ultimately, we can only ask whether a point lies on a line (incidence), or whether a point lies between two others on a line (ordering). The implied relations are formalised as:

$$\begin{aligned}\texttt{on\_line} &: \texttt{point} \to \texttt{line} \to \texttt{bool}\\ \texttt{between} &: \texttt{point} \to \texttt{point} \to \texttt{point} \to \texttt{bool}.\end{aligned}$$

These two primitives are then governed by five axioms[1]:

1. Two points determine exactly one line, and every line is determined by two points.
2. There is a triangle (three points that do not lie simultaneously on any given line).
3. If a point B lies between A and C then B lies between C and A, all three are distinct, collinear and A does not lie between B and C.
4. Given two points A and B, there is a point C such that B lies between A and C (line-segment extension).
5. If A, B and C form a triangle, and a line passes between two of its vertices, then it either passes between two other vertices, or passes through a vertex (Pasch's axiom).

We have formalised these axioms in HOL Light, and all results that follow in this paper have been formally verified from them[2]. Note that the theory we verified is actually a theory of three dimensional *space*, rather than the two dimensional plane. It is only in the interests of readability that we have dropped all mention of the planes on which our axioms and theorems are relativised.

The full verification comes in at around 2000 lines of proof code. It is written in a declarative style, with just about every step giving an intermediate result which follows from the preceding ones, and letting automation verify that it does so. Our chief pieces of automation are: first, the generic first-order theorem prover `MESON` [17]; second, a domain-specific search strategy for discharging side-conditions about incidence expressed using search combinators that we have described elsewhere [23]; and third, a proof command we implemented to reduce all linear incidence problems to linear arithmetic. The aggressive use of such generic automation trades verification speed for size. The proof is machine checked in about thirty minutes on an Intel Core 2 with a 2.53GHz clock speed, a reasonable price, we think, for having a verification that is relatively concise and readable, and where in many cases, it is possible to follow every step of the code, sketching the implied geometric figure as we go.

3.1 Elementary consequences

We have two crucial elementary theorems which we have verified from our axioms. Firstly, we have a linear ordering theorem on rays. To define a ray, we take an origin point O and a distinct point P with which to declare the ray as having direction $\overrightarrow{OP}$. The set of points on this ray are then the points between O and P, the point P itself, and the points X for which P is between O and X. These points are totally ordered by the relation $<_{OP}$ where $X <_{OP} Y$ holds when X is between O and Y. Moreover, the points on the ray are *densely* ordered. This is guaranteed by Hilbert's Theorem 3, which says that between any two points there exists another.

Secondly, we have a planar ordering theorem, which says that every line divides the plane into two regions such that two points in different regions are endpoints of a segment which intersects the line. Among other things, this theorem allows us to define the interior of a

[1]The formalised versions of these axioms are given in Appendix A.

[2]The verification code is available at https://github.com/Chattered/hilbert-bundle/tree/master/hol-light/hilbert.

triangle as the intersection of three interior sides (sides of an edge containing the opposite vertex). And from this, we can verify a theorem that is normally a consequence of the Crossbar Theorem and which we call the Interior Pasch Theorem:

Theorem (Interior Pasch)**.** *Any line through a vertex and an interior point of a triangle cuts the opposite edge.*

4 Polygons as Vertex Lists

The logic of HOL Light affords us an additional vocabulary in sets and lists, which we can use to define polygonal paths. In particular, we shall say that such a path is just a list of its vertices. Incidence with such paths is then defined in terms of incidence of points between adjacent vertices[3]:

$$\begin{aligned}
&\texttt{path\_contains} \;:\; \texttt{point list} \rightarrow \texttt{point} \rightarrow \texttt{bool}\\
\mathit{path}\;&\texttt{path\_contains}\; P\\
&\longleftrightarrow \texttt{mem}\; P\; \mathit{path} \;\vee\; (\exists X\; Y.\, \texttt{mem}\; (X,Y)\; (\texttt{adjacent}\; \mathit{path}) \;\wedge\; \texttt{between}\; X\; P\; Y)
\end{aligned}$$

A standard maze can then be defined as a closed polygonal path which does not self-intersect:

$$\begin{aligned}
&\texttt{standard\_maze} \;:\; \texttt{point list} \rightarrow \texttt{bool}\\
&\texttt{standard\_maze}\; \mathit{path}\\
&\longleftrightarrow\; 3 \leq \texttt{length}\; \mathit{path} \wedge \texttt{head}\; \mathit{path} = \texttt{last}\; \mathit{path}\\
&\quad \wedge \texttt{pairwise}\; (P\; Q \mapsto P \neq Q)\; (\texttt{init}\; \mathit{path})\\
&\quad \wedge (\forall P\; P'\; X.\, \texttt{mem}\; (P,P')\; (\texttt{adjacent}\; \mathit{path}) \wedge \texttt{between}\; P\; X\; P' \longrightarrow \neg\texttt{mem}\; X\; \mathit{path})\\
&\quad \wedge \texttt{pairwise}\; ((P,P')\; (Q,Q') \mapsto \neg(\exists X.\, \texttt{between}\; P\; X\; P' \wedge \texttt{between}\; Q\; X\; Q'))\\
&\qquad\qquad (\texttt{adjacent}\; \mathit{path})
\end{aligned}$$

4.1 The Problem Formalised

For us, navigating a maze means starting from an arbitrary point not on the maze (which we can formalise in terms of `path_contains` defined above) and finding a path which takes us around each of the maze walls without intersecting any of them. If we take our paths to be polygonal, then again, the absence of such intersections can be formalised in terms of `path_contains`.

We just need to define the relationship between each point and a maze wall, one which captures the idea of moving around the maze. A simple scheme we decided on is to say that each position around the maze is paired with a point on a maze wall, defining a line-of-sight. The obvious constraint on a line-of-sight is that the maze should not occlude it: there should be no other points along it which intersect the maze.

Our job then is to find a path, starting from a line-of-sight to a given wall of the maze, to a point with line-of-sight to another given wall. It will help to have a relation on points which

[3]See Appendix B for formal definitions of terms such as `adjacent`. Note that all free variables are implicitly universally quantified.

says they are joined by such a path, which we define by:

$$\begin{aligned}&\texttt{is\_connected\_to} \;:\; \texttt{point list} \to \texttt{point} \to \texttt{point} \to \texttt{bool}\\ &\texttt{is\_connected\_to}\ \mathit{figure}\ P\ Q\\ &\longleftrightarrow (\exists path.\ path \neq [\,] \ \wedge\ \texttt{head}\ path = P \ \wedge\ \texttt{last}\ path = Q\\ &\qquad \wedge\ \texttt{disjoint}\ (\texttt{path\_contains}\ path)\ \mathit{figure}).\end{aligned}$$

And now to our goal theorem:

$$\begin{aligned}&\texttt{standard\_maze}\ path\ \wedge\ \texttt{mem}\ (P_1, P_2)\ (\texttt{adjacent}\ path)\ \wedge\ \texttt{mem}\ (Q_1, Q_2)\ (\texttt{adjacent}\ path)\\ &\wedge\ \neg(path\ \texttt{path\_contains}\ A)\ \wedge\ \texttt{between}\ P_1\ A'\ P_2\\ &\wedge\ \neg(\exists X.\ \texttt{between}\ A\ X\ A'\ \wedge\ path\ \texttt{path\_contains}\ X)\\ &\longrightarrow\ (\exists B\ B'.\ \texttt{is\_connected\_to}\ (\texttt{path\_contains}\ path)\ A\ B\ \wedge\ \texttt{between}\ Q_1\ B'\ Q_2\\ &\qquad \wedge\ \neg(path\ \texttt{path\_contains}\ B)\ \wedge\ \neg(\exists X.\ \texttt{between}\ B\ X\ B'\ \wedge\ path\ \texttt{path\_contains}\ X)).\end{aligned}$$

4.2 Manipulating Vertex lists

Our standard mazes are represented as vertex lists with a common first and last element. Following a circuit around the maze therefore corresponds to rotating its vertex list, an idea that we can formalise as follows:

$$\begin{aligned}&\texttt{is\_rotation\_of} \;:\; \texttt{point list} \to \texttt{point list} \to \texttt{bool}\\ &ps\ \texttt{is\_rotation\_of}\ qs\\ &\quad\longleftrightarrow (\exists P\ Q\ ps'\ qs''.\\ &\qquad\qquad ps\ =\ [P]\ \texttt{++}\ ps'\ \texttt{++}\ [Q]\ \texttt{++}\ ps''\ \texttt{++}\ [P]\\ &\qquad\qquad \wedge\ qs\ =\ [Q]\ \texttt{++}\ ps''\ \texttt{++}\ [P]\ \texttt{++}\ ps'\ \texttt{++}\ [Q]\\ &\qquad\qquad \vee\ ps\ =\ qs).\end{aligned}$$

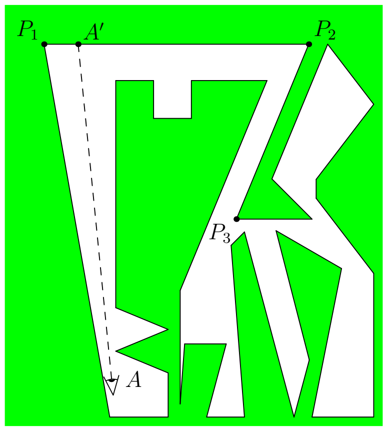
This relation is trivially reflexive and symmetric, and it is also provably transitive. What is more, all the properties which interest us, namely being a standard maze, being a member of the adjacency list, and being a point on a maze are invariant up to rotations of the vertex list. That is:

$$\begin{aligned}ps\ \texttt{is\_rotation\_of}\ qs\ \longrightarrow &(\texttt{mem}\ X\ (\texttt{adjacent}\ ps) \longleftrightarrow \texttt{mem}\ X\ (\texttt{adjacent}\ qs))\\ &\wedge (\texttt{standard\_maze}\ ps \longleftrightarrow \texttt{standard\_maze}\ qs)\\ &\wedge \texttt{path\_contains}\ ps\ =\ \texttt{path\_contains}\ qs\end{aligned}$$

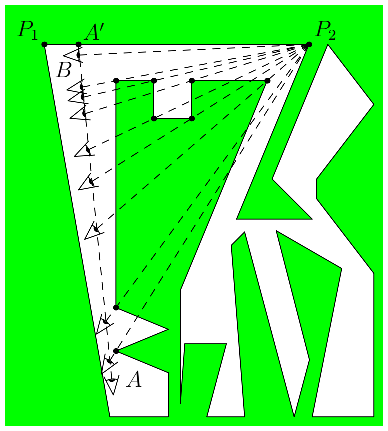
The relation and its invariances allow us to simplify our original problem. If we are at a random wall in a maze, then polygon rotations say that we can shift perspective, and imagine that we are instead at the *first* wall. This style of arguing without-loss-of-generality via invariance under a suitable equivalence class is well-known to the theorem proving community [10].

5 Navigating Standard Mazes

In this section, we will use a worked example to show the general strategy for navigating between the walls of a maze.



(a) Initial line-of-sight to P_1P_2



(b) Moving along AA'

Figure 1

5.1 The Convex Case

In Figure 1a, we have a point A with line-of-sight to the point A' on the wall P_1P_2. Our first move will take us to a point with a new line-of-sight to the wall P_2P_3.

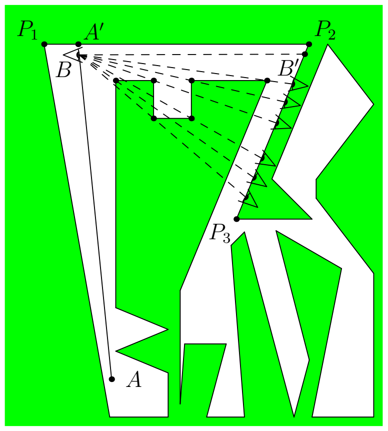
Appealing to the diagram, we would like to move ourselves towards A' so that P_2P_3 comes into view. We might consider using Theorem 3, which says that there is a point between any two distinct points. That would at least allow us to find a point between A and A' which, in some sense, takes us closer towards A'. But this point is arbitrary, and worse, it is possible that our line-of-sight is non-Archimedian and that we only ever move infinitesimally closer, never making progress even by repeated applications of Theorem 3. We need some other technique.

We first decide which point on P_2P_3 we aim to have a line-of-sight to. We aim initially for the vertex P_2, which we know must be visible somewhere along AA'. To move there, we will need to find suitable intersections along AA', and for this, we can use the Interior Pasch Theorem from §3.1.

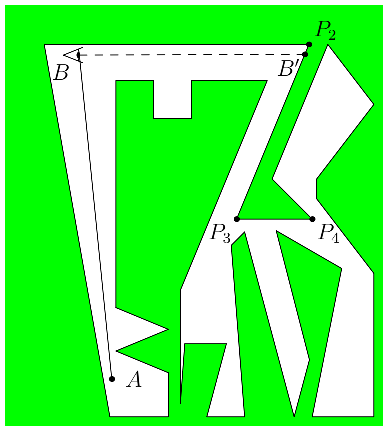
So let us take the triangle $AA'P_2$ and, for each vertex of the maze P which lies inside this triangle, we will find the point at which the line PP_2 intersects edge AA'. If we linearly order these intersections, we will find ourselves getting closer and closer to A', with fewer and fewer of the maze walls occluding our view of P_2. In Figure 1b, the very closest of these eight points marks the very last position along AA' where P_2 is still occluded. All we need do now is pick an arbitrary point between this final position and X', via Theorem 3, giving us our destination B, a point with a guaranteed unoccluded line-of-sight to P_2.

However, this is not sufficient. We need line-of-sight to a point on the *wall* P_2P_3. Most of this wall is still occluded by the maze, so we need some way to *rotate* ourselves by a sufficiently small fraction, adjusting our line-of-sight to a point on P_2P_3 very near the vertex P_2.

Now we managed to move ourselves to avoid occluding maze walls using our Interior Pasch Theorem, and we can perform the same trick in reverse, rotating ourselves so as to avoid the occluding maze walls. This time, we draw lines through every maze vertex inside the triangle BP_2P_3 to the edge P_2P_3. Taking the intersection closest to P_2, and applying Theorem 3 again, we find the target of a new line-of-sight: the point B' on P_2P_3, as in Figure 2a.



(a) Rotating to B'



(b) Line-of-sight to P_2P_3

Figure 2

5.2 The Concave Case

Starting at the point B in Figure 2b, we must obtain line-of-sight with the edge P_3P_4. This requires a slightly different strategy because, no matter how far along BB' we move, P_3P_4 will never come into view. This time, our target wall is on the opposite side of the wall we currently face (P_2P_3), with the vertex P_3 appearing to be a point of concavity. We will need a means to get round the corner of $P_2P_3P_4$.

We make two moves. First, we follow BB' until a point D just slightly past P_3 comes into view. This point D is obtained by casting a ray from P_3 in the direction P_2P_3 as in Figure 3a. Ray-casting is a useful technique available in ordered geometry, and can be performed by taking all intersections of a ray with a maze and picking the one closest to the ray's origin according to linear ordering. Formally, if X is an intersection of the ray OX with a maze given by *path*, we find the nearest such intersection Y:

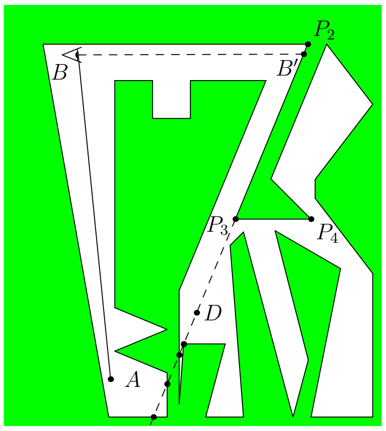
$$\begin{aligned} &\neg(\mathit{path}\ \texttt{path\_contains}\ O)\ \wedge\ \mathit{path}\ \texttt{path\_contains}\ X \\ &\quad\longrightarrow\ (\exists Y.\ \mathit{path}\ \texttt{path\_contains}\ Y\ \wedge\ (\texttt{between}\ O\ Y\ X\ \vee\ Y = X) \\ &\qquad\quad \wedge\ \neg(\exists Z.\ \texttt{between}\ O\ Z\ Y\ \wedge\ \mathit{path}\ \texttt{path\_contains}\ Z)) \end{aligned}$$

Still, our target D is occluded from our current position B by maze walls. So we must move along BB' by applying the same trick as before. This time, we draw lines through all points of the maze inside the triangle $BB'D$ to the edge BB'. See Figure 3b.

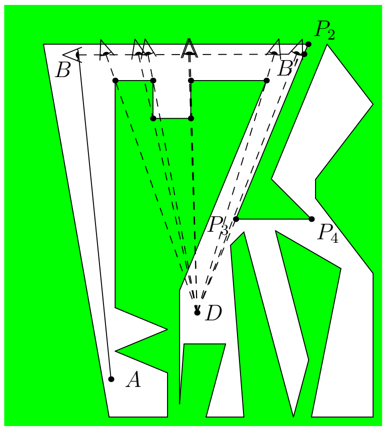
Since we have unoccluded sight to D, we can move there directly, and we notice that the wall P_3P_4 has come into view. But still, much of this wall is occluded. In Figure 4a, we show how to rotate ever-so-slightly to find our new line-of-sight DD', this time by considering intersections through the triangle P_3P_4D to the edge P_3P_4. We finally end up in the situation shown in Figure 4b.

For completeness, Figure 5 gives a final path through the entire maze[4].

[4]The diagrams are automatically generated by an implementation of the same algorithm we have verified.

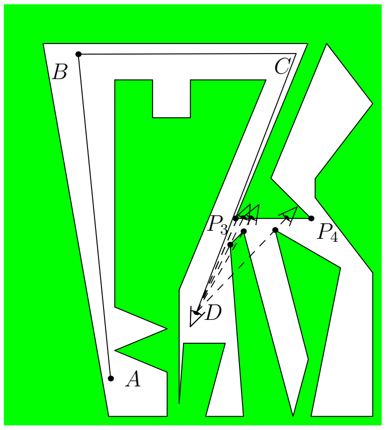


(a) Ray-cast to D

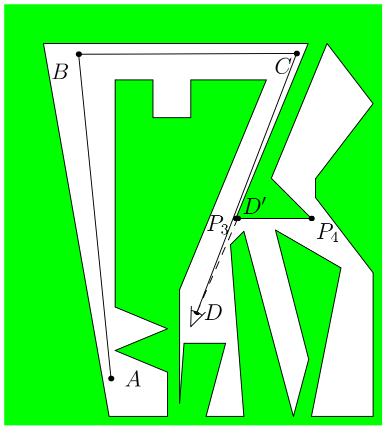


(b) Moving along BB'

Figure 3



(a) Turning to D'



(b) Line-of-sight to P_3P_4

Figure 4

5.3 Formalisation

Ordered geometry can be quite a misleading world, since diagrams imply constraints not given by the axioms and intuitive arguments can fail in surprising ways (as they did for Veblen). Some subtleties are revealed by formalisation which are not apparent in intuitive arguments. For instance, it is not immediately clear how a vertex being a point of concavity actually plays out in the argument and what assumptions are violated if we were to treat such points as we did with points of convexity. That the two scenarios must be treated differently is clear from diagrams, but how this works in terms of the basic ideas of ordered geometry is best understood by carefully inspecting the verification.

We can get an idea of this by considering the fairly complex formalisation of the central

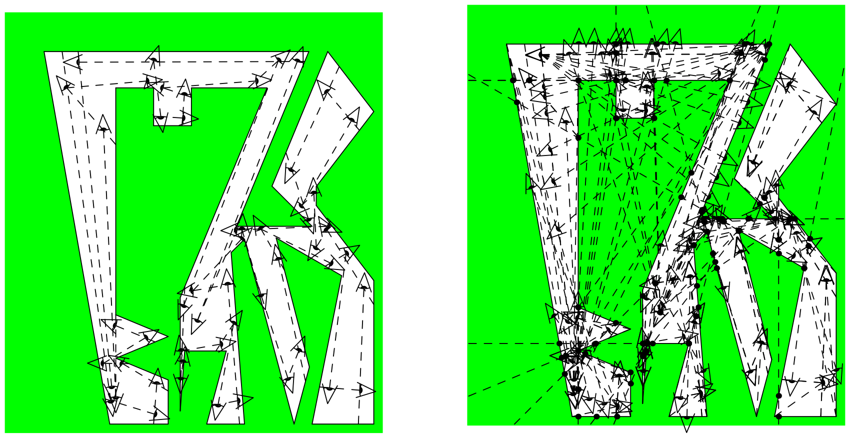

(a) Round the Maze (b) In Gory Detail

Figure 5

technique we used above to move and rotate towards new lines-of-sight:

$$\begin{aligned}&\neg(\exists a.\ P\ \texttt{on\_line}\ a\ \wedge\ X\ \texttt{on\_line}\ a\ \wedge\ Y'\ \texttt{on\_line}\ a)\\ &\wedge\ \neg(\mathit{path}\ \texttt{path\_contains}\ P)\ \wedge\ (\forall Z.\ \texttt{between}\ P\ Z\ X\ \longrightarrow\ \neg(\mathit{path}\ \texttt{path\_contains}\ Z))\\ &\wedge\ (\forall Z.\ \texttt{between}\ P\ Z\ Y'\ \longrightarrow\ \neg(\mathit{path}\ \texttt{path\_contains}\ Z))\\ &\longrightarrow\ (\exists Y.\ \texttt{between}\ P\ Y\ X\ \wedge\ (\forall Z.\ \texttt{between}\ Y'\ Z\ Y\ \longrightarrow\ \neg(\mathit{path}\ \texttt{path\_contains}\ Z)))\end{aligned}$$

It turns out that when using this theorem in verification, the variable `path` is never instantiated with the entire polygonal maze. Instead, we instantiate with the fragment of the maze obtained by dropping the first two points in its vertex list. This means that our conclusion does not guarantee us a genuine line-of-sight. We are left to finish the argument by proving that the discarded walls do not occlude the purported line-of-sight YY'. The argument can be somewhat fiddly, but ultimately boils down to reasoning about the two sides of the line which define the concavity or convexity.

More monstrous is the formalised theorem which actually says that we can move from line-of-sight with P_1P_2 to a line-of-sight with P_2P_3, a verified theorem which captures the full argument sketched in the last section:

$$\begin{aligned}&\texttt{between}\ P_1\ X'\ P_2\ \wedge\ P_2 \neq P_3\ \wedge\ \neg((P_1\ ::\ P_2\ ::\ P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ X)\\ &\wedge\ \neg((P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ P_2)\\ &\wedge\ \neg(\exists Z.\ \texttt{between}\ X\ Z\ X'\ \wedge\ (P_1\ ::\ P_2\ ::\ P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ Z)\\ &\wedge\ \neg(\exists Z.\ \texttt{between}\ P_1\ Z\ P_2\ \wedge\ (P_2\ ::\ P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ Z)\\ &\wedge\ \neg(\exists Z.\ \texttt{between}\ P_2\ Z\ P_3\ \wedge\ (P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ Z)\\ &\longrightarrow\ (\exists Y'\ Y.\ \texttt{is\_connected\_to}\ (\texttt{path\_contains}\ (P_1\ ::\ P_2\ ::\ P_3\ ::\ \mathit{path}))\ X\ Y\\ &\qquad\qquad \wedge\ \texttt{between}\ P_2\ Y'\ P_3\ \wedge\ \neg((P_1\ ::\ P_2\ ::\ P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ Y)\\ &\qquad\qquad \wedge\ \neg(\exists Z.\ \texttt{between}\ Y\ Z\ Y'\ \wedge\ (P_1\ ::\ P_2\ ::\ P_3\ ::\ \mathit{path})\ \texttt{path\_contains}\ Z))\end{aligned}$$

What is notable is that this theorem nowhere requires that the path $P_1 :: P_2 :: P_3 :: \mathit{path}$ defines a standard maze. The assumptions are a good deal weaker. We only need to know that

P_2 is distinct from P_3, that the line-of-sight XX' does not intersect the path, and that the two walls P_1P_2 and P_2P_3 do not intersect the path fragments which follow them. Those path fragments, notice, do not even need to be closed.

In our experience, the attention to detail required in verification, and the piecemeal way in which a verification might proceed by gradually adding hypotheses to a goal theorem as one becomes stuck, often means that we get theorems such as this which assume surprisingly very little.

In fact, this theorem is proven in our verification long before we have even *defined* a standard maze. The much stronger hypothesis that the path is such a maze only becomes needed as we start arbitrarily rotating the polygon as described in §4.2. Then, our initially weak hypotheses, focusing as they do on just a few walls of the maze, must be extended to make a general statement about *all* walls, and thus that the path is indeed a standard maze.

Still, the complexity of these theorems, with their numerous opaque hypotheses, is something to give us pause: it is all too easy to verify the wrong thing, and find oneself with a less than useful lemma. These complex conditionals are cumbersome. Every time we use them, their preconditions must be, sometimes painstakingly, verified. And there is always the possibility that some obscure pathology has been missed where the hypotheses just cannot be satisfied, and the proof is doomed to fail at the final hurdles. Fortunately for us, this was not the case.

6 Concluding Remarks

Even though Hilbert stated the theorem, and even though he was supposed to be working rigorously within ordered geometry, in a way which, according to Weyl [25], left "no gaps in the deductions", he nevertheless completely neglected to give a proof of it. This is very difficult to excuse when he provided full prose proofs for *far* simpler theorems. It seems Hilbert was just mistaken here, thinking that the theorem is easily proved. Veblen was more thorough, but despite his own credentials for mathematical rigour, his own proof has met scepticism.

We are left to wonder whether this is just a feature of the problem itself. We confess that, had we not completed a full verification of the theorem, we would not have been convinced of its truth. We had stumbled several times on intuitive proofs which turned out to be on very shaky logical footing, and even towards the end of the verification, we were seriously entertaining the possibility that there was a pathological non-Euclidean counterexample to the theorem.

In cases such as this, perhaps the devil is doomed to be in tortuous details which forever invite doubts about the correctness of any purported prose proof. When we restrict the axioms available to us, we are greatly generalising the scope and placing stronger demands on the theorem, requiring it to apply to quite alien geometries, of the sort where the simple pen and paper reasoning familiar to us simply does not apply.

As such, this may be a case where a full mechanical verification like ours is essential to securing a mathematical proof, where diagrams and intuitive arguments such as the one we gave in §5 must ultimately give way to fully symbolic proofs based on the sort of mechanical detail we gave in sections 4 and 5.3.

This is not to say that machine verified proof should *replace* prose arguments and diagrams. Such diagrams can at least help a human reader to understand the verification's approach and structure, and give the reader an intuitive sense of how the proof works, with the verification confirming that no question is begged nor unstated hypothesis presumed.

A Formalised Axioms

$$P \neq Q \longrightarrow (\exists a.\ P\ \texttt{on\_line}\ a \wedge Q\ \texttt{on\_line}\ a \wedge (\forall b.\ P\ \texttt{on\_line}\ b \wedge Q\ \texttt{on\_line}\ b \longrightarrow a = b)) \\ \wedge (\forall a.\ (\exists P\ Q.\ P \neq Q \wedge P\ \texttt{on\_line}\ a \wedge Q\ \texttt{on\_line}\ a)) \tag{1}$$

$$\exists P\ Q\ R.\ \neg(\exists a.\ P\ \texttt{on\_line}\ a \wedge Q\ \texttt{on\_line}\ a \wedge R\ \texttt{on\_line}\ a) \tag{2}$$

$$\begin{aligned} &\texttt{between}\ P\ Q\ R \\ &\quad \longrightarrow P \neq R \\ &\qquad \wedge (\exists a.\ P\ \texttt{on\_line}\ a \wedge Q\ \texttt{on\_line}\ a \wedge R\ \texttt{on\_line}\ a) \\ &\qquad \wedge \texttt{between}\ R\ Q\ P \\ &\qquad \wedge \neg\texttt{between}\ P\ R\ Q \end{aligned} \tag{3}$$

$$P \neq R \longrightarrow (\exists Q.\ \texttt{between}\ P\ R\ Q) \tag{4}$$

$$\begin{aligned} &\neg(\exists a.\ P\ \texttt{on\_line}\ a \wedge Q\ \texttt{on\_line}\ a \wedge R\ \texttt{on\_line}\ a) \\ &\quad \wedge \neg(P\ \texttt{on\_line}\ b) \wedge \neg(Q\ \texttt{on\_line}\ b) \wedge \neg(R\ \texttt{on\_line}\ b) \\ &\quad \wedge X\ \texttt{on\_line}\ b \wedge \texttt{between}\ P\ X\ Q \\ &\quad \longrightarrow (\exists Y.\ Y\ \texttt{on\_line}\ b \wedge (\texttt{between}\ P\ Y\ R \vee \texttt{between}\ Q\ Y\ R)) \end{aligned} \tag{5}$$

B Auxiliary definitions

The function $::$ is the most primitive function for lists, prepending elements.

$$\texttt{head}\ (x :: xs) = x \qquad \begin{aligned} \texttt{tail}\ [\,] &= [\,] \\ \texttt{tail}\ (x :: xs) &= xs \end{aligned}$$

$$\texttt{last}\ (x :: xs) = \texttt{if}\ (xs = [\,])\ \texttt{then}\ x\ \texttt{else}\ \texttt{last}\ xs$$

$$\begin{aligned} \texttt{init}\ [\,] &= [\,] \\ \texttt{init}\ (x :: xs) &= \texttt{if}\ (x = [\,])\ \texttt{then}\ [\,]\ \texttt{else}\ (x :: \texttt{init}\ xs) \end{aligned}$$

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

$$\begin{aligned} \texttt{mem}\ x\ [\,] &\longleftrightarrow \bot \\ \texttt{mem}\ x\ (y :: ys) &\longleftrightarrow x = y \vee \texttt{mem}\ x\ ys \end{aligned}$$

$$\begin{aligned} [\,] \mathbin{+\!\!+} xs &= xs \\ (x \mathbin{::} xs) \mathbin{+\!\!+} ys &= x \mathbin{::} (xs \mathbin{+\!\!+} ys) \end{aligned}$$

$$\texttt{adjacent}\ xs = \texttt{zip}\ (\texttt{init}\ xs)\ (\texttt{tail}\ xs)$$

$$\begin{aligned} \texttt{all}\ P\ [\,] &\longleftrightarrow \top \\ \texttt{all}\ P\ (x \mathbin{::} xs) &\longleftrightarrow P\ x \wedge \texttt{all}\ P\ xs \end{aligned}$$

$$\begin{aligned} \texttt{pairwise}\ r\ [\,] &\longleftrightarrow \top \\ \texttt{pairwise}\ r\ (x \mathbin{::} xs) &\longleftrightarrow \texttt{all}\ (r\ x)\ xs \wedge \texttt{pairwise}\ r\ xs \end{aligned}$$

References

[1] Gabriel Braun and Julien Narboux. From Tarski to Hilbert. In Tetsuo Ida and Jacques D. Fleuriot, editors, *Automated Deduction in Geometry*, volume 7993 of *Lecture Notes in Computer Science*, pages 89–109. Springer, 2013.

[2] 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.

[3] 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.

[4] 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.

[5] Georg Feigl. Über die elementaren Anordnungssätz. *Jahresbericht der Deutschen Mathematiker-Vereinigung*, 33, 1925.

[6] Heinrich W Guggenheimer. The Jordan and Schoefiles Theorems in Axiomatic Geometry. *The American Mathematical Monthly*, 85(9):pp. 753–756, 1978.

[7] Thomas C. Hales. Jordan's Proof of the Jordan Curve Theorem. *Studies in Logic, Grammar and Rhetoric*, 10(23), 2007.

[8] Thomas C. Hales. The Jordan Curve Theorem, Formally and Informally. *The American Mathematical Monthly*, 114:882–894, 2007.

[9] 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.

[10] 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.

[11] David Hilbert. *Foundations of Geometry.* Open Court Classics, 2nd edition, 1971. Translated from the 10th edition of the *Grundlagen der Geometrie*.

[12] T. Ida and J. Fleuriot, editors. *Proceedings of the 9th International Workshop on Automated Deduction in Geometry (ADG 2012)*, volume TBC of *Informatics Research Report*. University of Edinburgh, 2012. http://dream.inf.ed.ac.uk/events/adg2012/uploads/proceedings/ADG2012-proceedings.pdf.

[13] Camille Jordan. *Cours d'analyse de l'École polytechnique*. Gauthier-Villars et fils Paris, 1893.

[14] Vladimir Kanovei and Michael Reeken. A nonstandard proof of the Jordan curve theorem. *Pacific Journal of Mathematics*, 36(1):219–229, 1971.

[15] N. J. Lennes. Theorems on the Simple Finite Polygon and Polyhedron. *American Journal of Mathematics*, 33(1):pp. 37–62, 1911.

[16] Robert Vaughn Main. *A Critique of the Incidence and Order Axioms of Geometry*. PhD thesis, Oregon State University, 1970.

[17] Michael A. McRobbie and John K. Slaney, editors. *Optimizing proof search in model elimination*, volume 1104 of *Lecture Notes in Computer Science*. Springer, 1996.

[18] Laura I. Meikle and Jacques D. Fleuriot. Mechanical Theorem Proving in Computational Geometry. In *Automated Deduction in Geometry*, pages 1–18, 2004.

[19] Victor Pambuccian. The axiomatics of ordered geometry: I. Ordered incidence spaces. *Expositiones Mathematicae*, 29(1):24 – 66, 2011.

[20] Vesna Pavlovic Sana Stojanovic and Predrag Janicic. A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs. In *Automated Deduction in Geometry*, Lecture Notes in Computer Science, pages 201–220. Springer, 2010.

[21] Phil Scott. Mechanising Hilbert's *Foundations of Geometry* in Isabelle. Master's thesis, University of Edinburgh, 2008.

[22] 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.

[23] Phil Scott and Jacques D. Fleuriot. A Combinator Language for Theorem Discovery. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics - 11th International Conference*, volume 7362 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2012.

[24] Oswald Veblen. Theory on Plane Curves in Non-Metrical Analysis Situs. *Transactions of the American Mathematical Society*, 6(1):83–98, 1905.

[25] Hermann Weyl. David Hilbert and his mathematical work. *Bulletin of the American Mathematical Society*, 50:635, 1944.