Appendices for Automated Theorem Proving in Euler Diagram Systems

Gem Stapleton[1] Judith Masthoff [2] Jean Flower[1]
Andrew Fish[1] Jane Southern[1]

Technical Report VMG.06.2

[1]Visual Modelling Group, University of Brighton, Brighton, UK
{g.e.stapleton,andrew.fish,j.f.southern}@brighton.ac.uk

[2]University of Aberdeen, Aberdeen, UK.
jmasthoff@csd.abdn.ac.uk

1 Introduction

This report is a series of appendices to accompany the paper Automated Theorem Proving in Euler Diagram Systems. Here we include some details omitted from that paper and some additional discussions that may be of interest.

In appendix A, we give an overview of the A^* search algorithm in the context of theorem proving. We establish the expressiveness of Euler diagrams in appendix B. A complete worked example showing how to calculate the restrictive heuristic is given in appendix C. The proofs of the three theorems given in the paper are included in appendix D. The notion of clutter in Euler diagrams and how our proof writing rules steer Edith towards proofs containing diagrams with low 'clutter scores' is covered in appendix E. Details on how we generated proof tasks to evaluate Edith are given in appendix F. Finally, much of our evaluation is presented in appendix G, although the main results are included in the paper.

A Theorem Proving Using A*

Given a proof task (d_0, d_n), the A^* algorithm generates and stores a sequence of proof attempts. Initially, this sequence contains only a zero length proof attempt, namely $\langle d_0 \rangle$. Repeatedly, A^* removes the first proof attempt, P, from the sequence. If the last diagram in P is d_n, then a proof has been found. Otherwise, additional proof attempts are constructed by extending P, applying reasoning rules wherever possible to the last diagram in P. This creates many new proof attempts, each 1 longer than P, that are inserted into the stored sequence of proof attempts. The place in which they are inserted is derived from the sum of two functions, explained below.

One function, called the **heuristic**, estimates 'how far' the last diagram in the proof attempt is from the conclusion diagram. The other, called the **cost**, calculates 'how costly' it has been to reach the last diagram from the premise diagram. We define the cost of applying each reasoning rule to be 1. So, the cost of a proof attempt is precisely the length of the proof attempt. New proof attempts are inserted into the sequence, ordered in such a way that P_1 is before P_2 if P_1 has cost plus heuristic less than this sum for P_2.

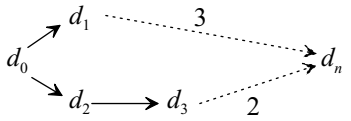

Figure 1: The sorting rule.

Example 1 In figure 1 there is an illustration of the type of search tree that A^* could generate. The diagram d_1 is 1 step away from the premise d_0, whereas d_3 is 2 steps away. It has been estimated, by a heuristic function, that d_1 is 3 steps away from the conclusion d_n whereas d_3 is estimated to be 2 steps away from d_n. Both proof attempts, $P_1 = \langle d_0, d_1 \rangle$ and $P_2 = \langle d_0, d_2, d_3 \rangle$, have cost plus heuristic equal to 4. If P_1 is ordered before P_2 by A^* then P_1 would be selected for extension before P_2. However, it seems reasonable to expect that a proof will be found more quickly by extending proof attempt P_2 since it is estimated to be only 2 steps away from a complete proof of d_n from d_0.

Suppose two distinct proof attempts $P_1 = \langle d_0, ..., d_i \rangle$ and $P_2 = \langle d_0, ..., d_j \rangle$ are generated by A^* when attempting to construct a proof of d_n from d_1. A **sorting rule** is incorporated into A^* as follows. If

$$cost(P_1) + heuristic(d_i, d_n) = cost(P_2) + heuristic(d_j, d_n)$$

then whenever $heuristic(d_i, d_n) < heuristic(d_j, d_n)$, P_1 is ordered before P_2. So, if the cost plus the heuristic is the same for two proof attempts then we order them according to the heuristic values. Thus, the sorting rule ensures that if P_1 and P_2 have the same cost plus heuristic value then we extend the proof attempt that we believe to be closer to a complete proof of the conclusion before extending the other proof attempt.

If $d_i = d_j$ then exactly one of P_1 and P_2 is stored by A^*. The proof attempt stored is the one with the lowest cost. In other words, P_1 is stored only if $cost(P_1) \leq cost(P_2)$, otherwise P_2 is stored. Currently in our implementation, if $cost(P_1) = cost(P_2)$ then the choice of which is stored by A^* is arbitrary.

The A^* algorithm always finds a solution with the lowest cost, if a solution exists, provided the heuristic used is *admissible* [3]. A heuristic is **admissible** if it never overestimates the cost of getting from any diagram to the conclusion diagram. A search algorithm that is guaranteed to find a path with the lowest cost is called **admissible**. Here, because each rule has cost 1, an admissible heuristic will provide a lower bound on the length of a shortest proof.

A.1 Why A*?

There are many reasons for choosing to utilize the A^* search algorithm when searching for an Euler diagram proof even if other algorithms may perform better from some perspectives. The performance of an algorithm can be measured in many ways, for example, time taken, space used and, specifically related to theorem proving, the lengths of the proofs produced. Moreover, in this setting, performance can also be measured in terms of the readability and accessibility of the resulting proofs. The length of a proof is an aspect of readability. Whilst shortest proofs are not necessarily the most readable, a short proof is desirable [2].

Using A^* allows us to be certain, given the admissibility of the heuristic, that a shortest proof will be found. This is one reason, but certainly not the only reason, that A^* is a good choice. A further reason for choosing A^* is because of its flexible nature: it is very easy to adapt A^* to incorporate user likes and dislikes, for example, by altering the cost function. Making some rules 'more costly' than others will allow Edith to produce proofs that minimise the number of rule applications which individual users find hard to understand, or simply dislike using. Thus Edith can be easily adapted to suit individual user preference, enhancing the readability and accessibility of the resulting proofs. In other words, A* can be tailored to suit user preference.

Further advantages of the flexible nature of A^* include the ability to steer Edith away from proofs that involve diagrams which Edith cannot au-

tomatically draw and, instead, presents at the abstract level. Again, the cost function can be altered to incorporate such functionality by making rule applications that result in undrawable diagrams more expensive.

The heuristics presented in the paper are still admissible when an increased cost function is used, so all of the theoretical results remain valid with these extensions. Since an aim of diagrammatic reasoning is to make proof reading (and writing) easier for non-expert users, such adaptations of the cost function have the potential to be important.

Hence, when considering performance more broadly than just considering time taken and space used, A^* is a reasonable choice of search algorithm.

B The Expressiveness of Euler Diagrams

Here we establish the expressiveness of Euler diagrams in terms of Monadic First Order Logic (MFOL). For example, the shading in d_1, figure 2, expresses that $B - A = \emptyset$ and corresponds to the universally quantified formula

$$\forall x \neg (B(x) \wedge \neg A(x)).$$

The fact that C is disjoint from $A \cup B$ is captured by

$$\forall x \neg \big((C(x) \wedge (A(x) \vee B(x)) \big).$$

The diagram d_1 is equivalent to the conjunction of these two MFOL sentences.

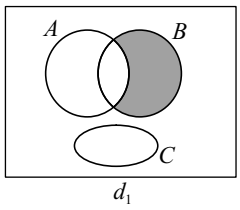

Figure 2: Converting Euler Diagrams into MFOL.

We seek a systematic way of classifying a fragment of MFOL that corresponds precisely to Euler diagrams. We note that Euler diagrams make statements about the emptiness of sets and nothing more. All of the elements in the universal set must lie in the sets represented by non-shaded zones. Thus, d_1 above is also equivalent to

$$\forall x \big((A(x) \wedge \neg B(x) \wedge \neg C(x)) \vee (A(x) \wedge B(x) \wedge \neg C(x)) \vee (\neg A(x) \wedge \neg B(x) \wedge C(x)) \vee (\neg A(x) \wedge \neg B(x) \wedge \neg C(x)) \big).$$

Definition 1 *Let f be a formula of MFOL, whose predicate symbols are drawn from $\mathcal{L}$, the contour labels used in Euler diagrams. Then f is in the* ***Euler diagram fragment*** *of MFOL if and only if f is satisfiable and of the form $\forall x\, g$ where the formula g contains no quantifiers and the only (free) variable in g is x.*

Lemma 1 *The Euler diagram logic is equivalent in expressive power to the Euler fragment of MFOL.*

C Calculating the Restrictive Euler Diagram Heuristic

In this appendix, we calculate the value of the restrictive heuristic function $EH_1(d_1, d_2)$ which is the sum of six difference measures for a particular example. Consider the proof task (d_1, d_2) where d_1 and d_2 are in figure 3. Calculating the two contour difference measures is our starting

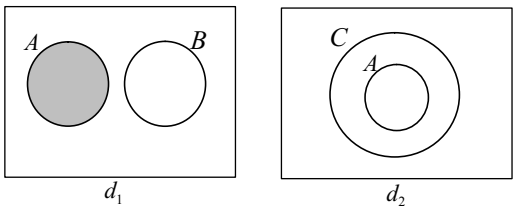

Figure 3: A proof task.

point. Since $L(d_2) - L(d_1) = \{C\}$, the add contour difference measure is $AddContour(d_1, d_2) = 1$. For the remove contour difference measure, we observe that $RemContour(d_1, d_2) = 1$ also, since $L(d_1) - L(d_2) = \{B\}$.

Now comes the more challenging task of computing the zone difference measures. First, we must create the contour forms of d_1 and d_2. These can be seen in figure 4 and are obtained by applying the restrictive add contour rule; C is added to d_1 and B is added to d_2 thus equalizing their contour label sets. The set $ZtoAdd(d_1, d_2)$ contains all of the zones that are in $CF(d_2, d_1)$ that are not in $CF(d_1, d_2)$. In this particular example, we have

$$ZtoAdd(d_1, d_2) = \{(A, B, C\}, \emptyset)\}.$$

Similarly,

$$ZtoRem(d_1, d_2) = \{(\{A\}, \{B, C\})\}.$$

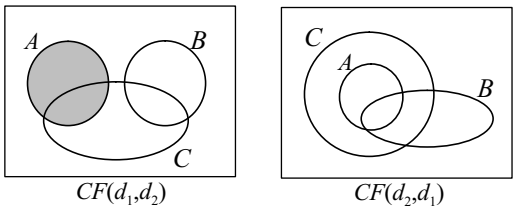

Figure 4: Contour forms.

Secondly, we identify the changeable zones of d_1 given d_2. These zones are those which are either missing or shaded in $CF(d_1, d_2)$, with the exception of the outside zone (which is neither missing nor, in this example, shaded). Therefore

$$ChZ(d_1, d_2) = \{(\{A, B\}, \{C\}), (\{A, B, C\}, \emptyset), (\{A\}, \{B, C\}), (\{A, C\}, \{B\})\}.$$

In order to compute the zone difference measures we need to find the span of the set $ChZ(d_1, d_2)$ given $d = CF(d_1, d_2)$. The span consists of the set of all zonal regions in d that are subsets of $ChZ(d_1, d_2)$. Whilst there are nine such zonal regions (as we will see), zones in $ChZ(d_1, d_2)$ immediately give rise to four of these zonal regions, namely

$$\langle\{A, B\}, \{C\}, d\rangle, \quad \langle\{A, B, C\}, \emptyset, d\rangle, \quad \langle\{A\}, \{B, C\}, d\rangle, \quad \langle\{A, C\}, \{B\}, d\rangle.$$

We can uses these four zonal regions to generate the remaining elements of $Span(ChZ(d_1, d_2))$ using the *combining rule*

$$\langle P \cup \{L\}, Q, d\rangle \cup \langle P, Q \cup \{L\}, d\rangle = \langle P, Q, d\rangle$$

where L is some label not in $P \cup Q$ (corresponding to the algebra of *splits* in [6]) as follows. We can take the union of $\langle\{A, B, C\}, \emptyset, d\rangle$ and $\langle\{A, B\}, \{C\}, d\rangle$ to give $\langle\{A, B\}, \emptyset, d\rangle$; this zonal region is shaded in d_3, figure 5. Similarly we

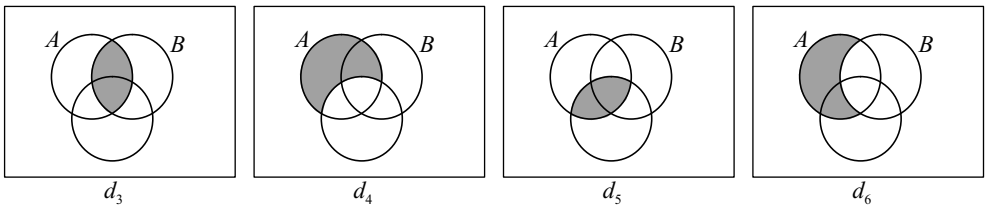

Figure 5: Computing spans.

have

$$\begin{aligned}
\langle\{A,B\},\{C\},d\rangle \cup \langle\{A\},\{B,C\},d\rangle &= \langle\{A\},\{C\},d\rangle \\
\langle\{A,B,C\},\emptyset,d\rangle \cup \langle\{A,C\},\{B\},d\rangle &= \langle\{A,C\},\emptyset,d\rangle \\
\langle\{A,C\},\{B\},d\rangle \cup \langle\{A\},\{B,C\},d\rangle &= \langle\{A\},\{B\},d\rangle
\end{aligned}$$

which are shaded in d_4, d_5 and d_6 respectively. Observing that a zonal region, $\langle P,Q,d\rangle$ contains 2^n zones, where $n = |L(d) - (P \cup Q)|$, we do not need to seek zonal regions containing three zones. Having found zonal regions that contain either one zone or two zones, our next step is to take unions of zonal regions that contain two zones each, giving zonal regions that contain four zones each:

$$\begin{aligned}
\langle\{A,B\},\emptyset,d\rangle \cup \langle\{A\},\{B\},d\rangle &= \langle\{A\},\emptyset,d\rangle \\
\langle\{A,C\},\emptyset,d\rangle \cup \langle\{A\},\{C\},d\rangle &= \langle\{A\},\emptyset,d\rangle.
\end{aligned}$$

Continuing this process, we now determine whether we can combine zonal regions containing four zones each to give zonal regions containing eight zones each. We have only one zonal region containing four zones, so there is no zonal region containing eight zones in $Span(ChZ(d_1,d_2))$. Thus we have created all of the zonal regions in $Span(ChZ(d_1,d_2))$:

$$\begin{aligned}
Span(ChZ(d_1,d_2)) = \{&\langle\{A,B\},\{C\},d\rangle, \langle\{A,B,C\},\emptyset,d\rangle, \langle\{A\},\{B,C\},d\rangle, \\
&\langle\{A,C\},\{B\},d\rangle, \langle\{A,B\},\emptyset,d\rangle, \langle\{A\},\{C\},d\rangle, \\
&\langle\{A,C\},\emptyset,d\rangle, \langle\{A\},\{B\},d\rangle, \langle\{A\},\emptyset,d\rangle\}.
\end{aligned}$$

A general algorithm to compute the span of a set of zones is given at the end of this section.

Having obtained $ZtoAdd(d_1,d_2) = \{(A,B,C\},\emptyset)\}$, $ZtoRem(d_1,d_2) = \{(\{A\},\{B,C\})\}$ and $Span(ChZ(d_1,d_2))$, we are in a position to compute the zone difference measures. In this particular example, each difference measure takes the value of 1 since, for the add zone difference measure, $ZtoAdd(d_1,d_2) = \{(A,B,C\},\emptyset)\}$ is covered by $\{\langle(A,B,C\},\emptyset,d\rangle\}$ and, for the remove zone difference measure, $ZtoRem(d_1,d_2) = \{(\{A\},\{B,C\})\}$ is covered by $\{\langle\{A\},\{B,C\},d\rangle\}$.

The next step in our example is to compute the shading difference measures. The first stage is to create the Venn forms of $CF(d_1,d_2)$ and $CF(d_2,d_1)$. This is done by adding shaded zones to the contour forms. To $CF(d_1,d_2)$, we can add the missing zone $(\{A,B\},\{C\})$ giving d_7 in figure 6. Next, add the missing zone $(\{A,B,C\},\emptyset)$ to d_7 giving $Venn(CF(d_1,d_2))$. A similar process for $CF(d_2,d_1)$ can be seen in figure 7. Using the Venn forms, we establish

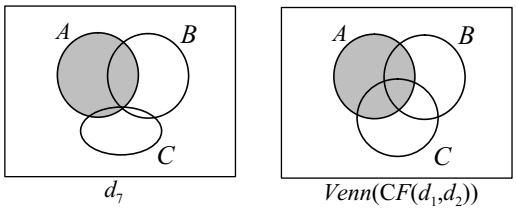

Figure 6: Obtaining the Venn form of $CF(d_1, d_2)$.

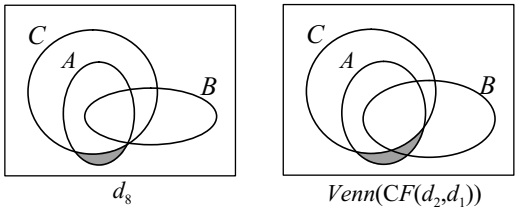

Figure 7: Obtaining the Venn form of $CF(d_2, d_1)$.

$ShdtoAdd(d_1, d_2) = \emptyset$ and $ShdtoRem(d_1, d_2) = \{(\{A, C\}, \{B\}), (\{A, B, C\}, \emptyset)\}$. Clearly $AddShading(d_1, d_2) = 0$ because there is no shading to be added. To compute $RemShading(d_1, d_2)$, we need to find $Span(ShdtoRem(d_1, d_2))$. Following the same process as above to find a span, we obtain

$$Span(ShdtoRem(d_1, d_2)) = \{\langle\{A, C\}, \{B\}, d\rangle, \langle\{A, B, C\}, \emptyset, d\rangle, \langle\{A, C\}, \emptyset, d\rangle\}.$$

Therefore, $RemShading(d_1, d_2) = 1$, since $ShdtoRem(d_1, d_2)$ is covered by $\{\langle\{A, C\}, \emptyset, d\rangle\}$.

The restrictive heuristic is the sum of the six difference measures, so

$$EH_1(d_1, d_2) = 1 + 1 + 1 + 1 + 0 + 1 = 5.$$

Hence a shortest proof of d_2 from d_1 is at least 5 steps long (in this example, a shortest proof requires exactly 5 steps).

In general, the task of computing covering numbers is analogous to the graph theoretic problem of computing a *dominating number*, a well-known NP-hard problem [5].

Definition 2 *Let G be a bipartite graph with partite sets U and V. Let X be a subset of V. Set X is said to* ***dominate*** *U if and only if each vertex in U is adjacent to some vertex in X. The cardinality of a smallest such X is called the* ***dominating number*** *for U in G.*

In our case, we can take the set of zones to be covered as U and the set $Span(ChZ(d_1, d_2))$ as V. There is an edge between zone $z \in U$ and

zonal region $zr \in V$ if and only if $z \in zr$, giving a bipartite graph G. The dominating number for U in G is then the same as the covering number for U with respect to $V = Span(ChZ(d_1, d_2))$.

In general, to compute $Span(ChZ(d_1, d_2))$ given diagrams d_1 and d_2, apply the following algorithm (where we denote $CF(d_1, d_2)$ by d):

Step 1 If $ChZ(d_1, d_2) = \emptyset$ then $Span(ChZ(d_1, d_2)) = \emptyset$ and terminate. Otherwise proceed to step 2.

Step 2 Set $i = 1$ and create a list

$$CompSpan_i = (\langle P_{1,i}, Q_{1,i}, d\rangle, ..., \langle P_{m,i}, Q_{m,i}, d\rangle)$$

where $ChZ(d_1, d_2) = \{(P_1, Q_1), ..., (P_m, Q_m)\}$ and, for each n where $1 \leq n \leq m$,

$$\langle P_{n,i}, Q_{n,i}, d\rangle = \{(P_n, Q_n)\}.$$

This gives the zonal regions arising from the zones in $ChZ(d_1, d_2)$.

Step 3 Set $j = 1$, $k = 2$, $l = 1$ and $CompSpan_{i+1} = ()$.

Step 4 Determine whether the combining rule can be applied to $\langle P_{j,i}, Q_{j,i}, d\rangle$ and $\langle P_{k,i}, Q_{k,i}, d\rangle$. If the rule can be applied then add their union, called $\langle P_{l,i+1}, Q_{l,i+1}, d\rangle$, to the end of the list $CompSpan_{i+1}$ and increment both l and k by 1. Otherwise, increment only k by 1.

Step 5 If k is less than or equal to the length of $CompSpan_i$ then return to step 4. Otherwise increment j by 1.

Step 6 If j is less than the length of $CompSpan_i$ then set $k = j + 1$ and return to step 4. Otherwise proceed to step 7.

Step 7 If $CompSpan_{i+1}$ is not the empty list then increment i by 1 and return to step 3. Otherwise, the elements of $Span(ChZ(d_1, d_2))$ are precisely those zonal regions that occur in any of the $CompSpan_i$ lists created above and we terminate.

Of course, there is a proof obligation that the given algorithm does indeed compute $Span(ChZ(d_1, d_2))$. Clearly, any zonal region that occurs in any of the $CompSpan_i$ lists is an element of $Span(ChZ(d_1, d_2))$ since it is created by taking unions of zonal regions that originate from zones in $ChZ(d_1, d_2)$ (a more formal argument would proceed by induction on i). Conversely, let $\langle P, Q, d\rangle$ be an element of $Span(ChZ(d_1, d_2))$. If $\langle P, Q, d\rangle$ contains just a single zone then $\langle P, Q, d\rangle$ is in $CompSpan_1$ and, we note, $|L(d) - (P \cup Q)| = 0$.

Assume that, if $|L(d)-(P\cup Q)| = k$ then $\langle P, Q, d\rangle$ is in the list $CompSpan_{k+1}$. Suppose that $\langle P, Q, d\rangle$ satisfies $|L(d) - (P \cup Q)| = k + 1$. Choose some label, L say, in $L(d) - (P \cup Q)$. Then

$$\langle P, Q, d\rangle = \langle P \cup \{L\}, Q, d\rangle \cup \langle P, Q \cup \{L\}, d\rangle.$$

By assumption, $\langle P \cup \{L\}, Q, d\rangle$ and $\langle P, Q \cup \{L\}, d\rangle$ are in $CompSpan_{k+1}$. It is then clear that $\langle P, Q, d\rangle$ is in $CompSpan_{k+2}$, as required.

D Proofs of Theorems 1,2 and 3

The lemma below is used in the proof of theorem 1.

Lemma 2 *Let d_1 and d_2 be Euler diagrams. If $EH_1(d_1, d_2) = \infty$ then $AddShading(d_1, d_2) = \infty$.*

Proof Suppose that $EH_1(d_1, d_2) = \infty$. Then either the add shading or the remove zone difference measure is infinite. Suppose that the remove zone difference measure is infinite. Then there is a zone, z say, in $ZtoRem(d_1, d_2)$ that is not in $ChZ(d_1, d_2)$. This means that z is not shaded in $CF(d_1, d_2)$ and, therefore, not shaded in $Venn(CF(d_1, d_2))$. Now, since z is in $RemZ(d_1, d_2)$, it must be that z is missing from the diagram $CF(d_2, d_1)$. Therefore, z is shaded in $Venn(CF(d_2, d_1))$, so the add shading difference measure is also infinite.

Theorem 1 *Let d_1 and d_2 be Euler diagrams. Then $EH_1(d_1, d_2) = \infty$ if and only if $d_1 \nvdash d_2$.*

Proof Suppose that $EH_1(d_1, d_2) = \infty$, so $AddShading(d_1, d_2) = \infty$. Therefore there is a zone that is shaded in $Venn(CF(d_2, d_1))$ but not shaded in $Venn(CF(d_1, d_2))$. Choose such a zone, z. It can be shown that $Venn(CF(d_1, d_2))$ is semantically equivalent to d_1 and $Venn(CF(d_2, d_1))$ is semantically equivalent to d_2. Now, in any model for $Venn(CF(d_2, d_1))$, the shaded zone z represents the empty set. It is simple to construct a model for $Venn(CF(d_1, d_2))$ in which z does not represent the empty set because z is not shaded in $Venn(CF(d_1, d_2))$. Therefore

$$Venn(CF(d_1, d_2)) \nvdash Venn(CF(d_2, d_1)).$$

If follows that $d_1 \nvdash d_2$. By soundness $d_1 \nvdash d_2$. Hence, if $EH_1(d_1, d_2) = \infty$ then $d_1 \nvdash d_2$.

Conversely, suppose that $d_1 \nvdash d_2$. Then, by completeness, $d_1 \nvDash d_2$. Therefore

$$Venn(CF(d_1, d_2)) \nvDash Venn(CF(d_2, d_1)).$$

Thus there is a model, m, for $Venn(CF(d_1, d_2))$ that is not a model for $Venn(CF(d_2, d_1))$. Choose such a model. Since there are no missing zones in $Venn(CF(d_2, d_1))$, the only way m fails to be a model for $Venn(CF(d_2, d_1))$ is because there exists a zone, z say, that is shaded in $Venn(CF(d_2, d_1))$ but does not represent the empty set. Choose such a zone, z. Since m is a model for $Venn(CF(d_1, d_2))$ it follows that z is not shaded in $Venn(CF(d_1, d_2))$. Therefore $AddShading(d_1, d_2) = \infty$. Hence, $EH_1(d_1, d_2) = \infty$ if and only if $d_1 \nvdash d_2$

Theorem 2 *The heuristic function for the restrictive Euler diagram system is admissible. That is, for all Euler diagrams d_1 and d_2, $EH_1(d_1, d_2)$ provides a lower bound on the length of a shortest proof of d_2 from d_1.*

Proof The proof is by induction on the shortest proof length. For the base case, we consider proofs of length $n = 0$. In this case, the premise d_0 is also the conclusion and it is trivial that $EH_1(d_0, d_0) = 0$. Hence the base case holds.

Assume that, for all Euler diagrams d_0 and d_k with shortest proof length $n = k$ of d_k from d_0, $EH_1(d_0, d_k) \leq k$.

Let d_0 and d_{k+1} be Euler diagrams with shortest proof length $k + 1$ of d_{k+1} from d_0. Choose a shortest proof, say $\langle d_0, d_1, d_2, ..., d_{k+1} \rangle$. Then $\langle d_1, d_2, ..., d_k, d_{k+1} \rangle$ is a shortest proof of d_{k+1} from d_1 and has length k, see figure 8. By assumption $EH_1(d_1, d_{k+1}) \leq k$. We want to show that

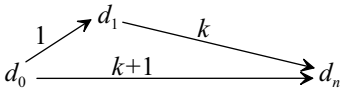

Figure 8: The proof strategy for theorem 2.

$EH_1(d_0, d_{k+1}) \leq k+1$. If we can show that $EH_1(d_0, d_{k+1}) \leq EH_1(d_1, d_{k+1})+1$ then, since we know $EH_1(d_1, d_{k+1}) \leq k$ it will follow that $EH_1(d_0, d_{k+1}) \leq k + 1$. There are five rules that could have been applied to d_0 giving d_1. The strategy is to consider each of the rules in turn and consider how applying the rule affects the values of the difference measures. We include the case when the remove shading from a zone rule has been applied to d_0, removing the shading from the zone (a, b). The remaining cases are more straightforward.

When we apply the remove shading from a zone rule to d_0 giving d_1 we do not change the contour label set, that is $L(d_0) = L(d_1)$ and we immediately have the following relationships between three of the difference measures:

(a) $AddContour(d_0, d_{k+1}) = AddContour(d_1, d_{k+1})$,

(b) $RemContour(d_0, d_{k+1}) = RemContour(d_1, d_{k+1})$ and

(c) $AddShading(d_0, d_{k+1}) = AddShading(d_1, d_{k+1}) = 0$ (by theorem 1).

Furthermore, because the zone sets of d_0 and d_1 are identical (we have only removed shading), their contour forms with respect to d_{k+1} also have identical zones sets. Therefore

$$ZtoAdd(d_0, d_{k+1}) = ZtoAdd(d_1, d_{k+1}) \qquad (1).$$

Since shading has been erased from d_0 to give d_1, the shaded zones in $CF(d_1, d_{k+1})$ form a proper subset of those which are shaded in $CF(d_0, d_{k+1})$. Furthermore, since the zones sets of the contour forms are identical, the contour forms also have the same missing zone sets. Therefore, the changeable zone sets satisfy

$$ChZ(d_1, d_{k+1}) \subset ChZ(d_0, d_{k+1}),$$

which implies that every zonal region which is a subset of $ChZ(d_1, d_{k+1})$ is also a subset of $ChZ(d_0, d_{k+1})$, in other words

$$Span(ChZ(d_1, d_{k+1})) \subset Span(ChZ(d_0, d_{k+1})) \qquad (2).$$

Choose a set of zonal regions, ZR say, satisfying

$$ZR \subseteq Span(ChZ(d_1, d_{k+1}))$$

which is a smallest covering of $ZtoAdd(d_1, d_{k+1})$, given $Span(ChZ(d_1, d_{k+1}))$. By (1), ZR is also a covering of $ZtoAdd(d_0, d_{k+1})$. By (2)

$$ZR \subseteq Span(ChZ(d_0, d_{k+1})).$$

Hence, given $Span(ChZ(d_0, d_{k+1}))$, either ZR is a smallest covering of

$$ZtoAdd(d_0, d_{k+1})$$

or there is a smaller covering and we deduce that

$$AddZone(d_0, d_{k+1}) \leq AddZone(d_1, d_{k+1}).$$

Similarly,

$$RemZone(d_0, d_{k+1}) \leq RemZone(d_1, d_{k+1}).$$

Finally, we consider the remove shading measure. Now, because every zone that is shaded in $CF(d_1, d_{k+1})$ is also shaded in $CF(d_0, d_{k+1})$ and we have applied the remove shading rule to d_0 giving d_1 we deduce that

$$ShdtoRem(d_1, d_{k+1}) \subset ShdtoRem(d_0, d_{k+1}).$$

Recall that shading was removed from the zone (a, b) in d_0 to give d_1. The zones in the region

$$R = ShdtoRem(d_0, d_{k+1}) - ShdtoRem(d_1, d_{k+1}) \qquad (3)$$

are precisely those which are shaded in $Venn(CF(d_0, d_{k+1}))$ but not shaded in $Venn(CF(d_1, d_{k+1}))$ and arise from the shading being removed from (a, b). The region R is, therefore, a zonal region given a and b. More formally, R is the zonal region $\langle a, b, Venn(CF(d_1, d_{k+1}))\rangle$. Therefore, by (3)

$$ShdtoRem(d_1, d_{k+1}) \cup \langle a, b, Venn(CF(d_1, d_{k+1}))\rangle = ShdtoRem(d_0, d_{k+1}) \quad (4).$$

Choose a set of zonal regions, ZR say, satisfying

$$ZR \subseteq Span(ShdtoRem(d_1, d_{k+1}))$$

which is a smallest covering of $ShdtoRem(d_1, d_{k+1})$. By (4),

$$ZR \cup \{\langle a, b, Venn(CF(d_0, d_{k+1}))\rangle\}$$

covers $ShdtoRem(d_0, d_{k+1})$. Hence we require at most one more zonal region to cover $ShdtoRem(d_0, d_{k+1})$ than to cover $ShdtoRem(d_1, d_{k+1})$, that is

$$RemShading(d_0, d_{k+1}) \leq RemShading(d_1, d_{k+1}) + 1.$$

We have shown that the difference measures between d_0 and d_{k+1} are either the same or are smaller than those between d_1 and d_{k+1} with the exception of the remove shading measure which can be at most 1 bigger. Therefore, since the heuristic function is the sum of these measures, $EH_1(d_0, d_{k+1}) \leq EH_1(d_1, d_{k+1}) + 1$. No matter which rule was applied to d_0 to give d_1, it can be shown that

$$\begin{aligned} EH_1(d_0, d_{k+1}) &\leq EH_1(d_1, d_{k+1}) + 1 \\ &\leq k + 1 \end{aligned}$$

Hence $EH_1(d_1, d_2)$ is admissible.

Theorem 3 *The heuristic function for the relaxed Euler diagram system is admissible. That is, for all Euler diagrams d_1 and d_2, $EH_2(d_1, d_2)$ provides a lower bound on the length of a shortest proof of d_2 from d_1.*

Proof Following the strategy of theorem 2, the proof is by induction on the shortest proof length. For the base case, we consider proofs of length $n = 0$. In this case, the premise d_0 is also the conclusion and it is trivial that $EH_2(d_0, d_0) = 0$. Hence the base case holds.

Assume that, for all Euler diagrams d_0 and d_k with shortest proof length $n = k$ of d_k from d_0, $EH_2(d_0, d_k) \leq k$.

Let d_0 and d_{k+1} be Euler diagrams with shortest proof length $k + 1$ of d_{k+1} from d_0. Choose a shortest proof, say $\langle d_0, d_1, d_2, ..., d_{k+1}\rangle$. Then $\langle d_1, d_2, ..., d_k, d_{k+1}\rangle$ is a shortest proof of d_{k+1} from d_1 and has length k.

By assumption $EH_2(d_1, d_{k+1}) \leq k$. We want to show that $EH_2(d_0, d_{k+1}) \leq k + 1$. There are five relaxed rules that could have been applied to d_0 giving d_1. The strategy is to consider each of the rules in turn and consider how applying the rule affects the values of the relaxed difference measures. We include the case when the relaxed remove a contour rule has been applied to d_0, removing the contour c. The remaining cases are more straightforward.

When we apply the relaxed remove a contour rule to d_0 removing c to give d_1, the contour label set is changed and $L(d_0) = L(d_1) \cup \{c\}$. There are two subcases to consider: either c is in $L(d_{k+1})$ or it is not.

Firstly, suppose that $c \notin L(d_{k+1})$. Then we immediately have the following relationships between three of the difference measures:

(a) $AddContour(d_0, d_{k+1}) = AddContour(d_1, d_{k+1})$,

(b) $RemContour(d_0, d_{k+1}) - 1 = RemContour(d_1, d_{k+1})$ and

(c) $AddShading(d_0, d_{k+1}) = AddShading(d_1, d_{k+1}) = 0$ (by theorem 1).

It is obvious that $RelCF(d_0, d_{k+1}) = RelCF(d_1, d_{k+1})$ and $RelCF(d_{k+1}, d_0) = RelCF(d_{k+1}, d_1)$ so

(a) $AddRegion(d_0, d_{k+1}) = AddRegion(d_1, d_{k+1})$,

(b) $RemRegion(d_0, d_{k+1}) = RemRegion(d_1, d_{k+1})$ and

(c) $RelRemShading(d_0, d_{k+1}) = RelRemShading(d_1, d_{k+1})$

Hence, in the case where $c \notin L(d_{k+1})$, we see that the sum of the difference measures between d_0 and d_{k+1} is exactly one more than the sum between d_1 and d_{k+1}. It immediately follows that $EH_2(d_0, d_{k+1}) \leq EH_2(d_1, d_{k+1}) + 1$.

Alternatively, in the more challenging case, $c \in L(d_{k+1})$. Then we immediately have the following relationships between three of the difference measures:

(a) $AddContour(d_0, d_{k+1}) + 1 = AddContour(d_1, d_{k+1})$,

(b) $RemContour(d_0, d_{k+1}) = RemContour(d_1, d_{k+1})$ and

(c) $AddShading(d_0, d_{k+1}) = AddShading(d_1, d_{k+1}) = 0$ (by theorem 1).

The remaining three measures can only take the values of 0 or 1. Thus the only way $EH_2(d_0, d_{k+1})) \leq EH_2(d_1, d_{k+1}) + 1$ can fail to hold is when

$$AddRegion(d_1, d_{k+1}) = RemRegion(d_1, d_{k+1}) = RelRemShading(d_1, d_{k+1}) = 0$$

and

$$AddRegion(d_0, d_{k+1}) = RemRegion(d_0, d_{k+1}) = RelRemShading(d_0, d_{k+1}) = 1.$$

We therefore only need to consider this scenario, from which we deduce that the sum of the difference measures between d_0 and d_{k+1} is

$$AddContour(d_0, d_{k+1}) + RemContour(d_0, d_{k+1}) + 3 \quad (1)$$

The shortest proof $\langle d_0, d_1, d_2, ..., d_{k+1} \rangle$ involves adding each contour that contributes to $AddContour(d_0, d_{k+1})$, and removing each contour that contributes to $RemContour(d_0, d_{k+1})$; this gives

$$AddContour(d_0, d_{k+1}) + RemContour(d_0, d_{k+1})$$

steps. Furthermore, the first step of this shortest proof removes c and, because c is in $L(d_{k+1})$, a further proof step must add c. Therefore, the length, $k + 1$, of this shortest proof satisfies

$$k + 1 \geq AddContour(d_0, d_{k+1}) + RemContour(d_0, d_{k+1}) + 2.$$

We observe that the only way (1) is not less than or equal to $k + 1$ is when

$$k + 1 = AddContour(d_0, d_{k+1}) + RemContour(d_0, d_{k+1}) + 2 \quad (2).$$

Therefore, we assume that (2) is, indeed, the case. It is then obvious that the proof $\langle d_0, d_1, d_2, ..., d_{k+1} \rangle$ uses only add and remove contour rules.

Recall that $EH_2(d_0, d_{k+1})$ is either (1) or one less than (1) (when we are in the 'contour correctable' case). For $EH_2(d_0, d_{k+1})$ to provide a lower bound on shortest proof length, $EH_2(d_0, d_{k+1})$ must be one less than (1)

and, therefore, satisfy $EH_2(d_0, d_{k+1}) = k + 1$. Now $EH_2(d_0, d_{k+1}) = k + 1$ when $RelCF(d_0, d_{k+1})$ is contour correctable given $RelCF(d_{k+1}, d_0)$. To show we are in the contour correctable case, we must find a contour, l say, in $RelCF(d_0, d_{k+1})$ whose removal from, and subsequent addition to, $RelCF(d_0, d_{k+1})$ results in $RelCF(d_{k+1}, d_0)$. We will show that taking $l = c$ establishes that the contour correctable relation holds.

Removing c from $RelCF(d_0, d_{k+1})$ yields $RelCF(d_1, d_{k+1})$. We know that $RelCF(d_1, d_{k+1}) = RelCF(d_{k+1}, d_1)$ since there are no region differences and no shading differences between these two contour forms (the four measures relevant to these differences all have value 0 from d_1 to d_{k+1} by assumption). Thus we must show that it is possible to add c to $RelCF(d_{k+1}, d_1)$ in order to obtain $RelCF(d_{k+1}, d_0)$, illustrated by

$$\begin{array}{ccc} RelCF(d_{k+1}, d_0) & \xleftarrow{\text{add } c} & RelCF(d_{k+1}, d_1) \\ \text{contour correctable} \uparrow & & \| \\ RelCF(d_0, d_{k+1}) & \xrightarrow[\text{remove } c]{} & RelCF(d_1, d_{k+1}) \end{array}$$

Now $RelCF(d_{k+1}, d_1)$ is obtained from $RelCF(d_{k+1}, d_0)$ by removing c. If we can show that removing c from $RelCF(d_{k+1}, d_0)$ does not lose any shading then we are able to add c to $RelCF(d_{k+1}, d_1)$ obtaining $RelCF(d_{k+1}, d_0)$. So, we show that $RelCF(d_{k+1}, d_0)$ is semantically equivalent to $RelCF(d_{k_+1}, d_1)$ since the only way this can fail to be the case is when shading is lost on the removal of c from $RelCF(d_{k+1}, d_0)$. It is straightforward to show that removing c from $RelCF(d_{k+1}, d_0)$ preserves information if and only if removing c from d_{k+1} preserves information. Thus we prove that removing c from d_{k+1} preserves information.

The proof $\langle d_0, d_1, d_2, ..., d_{k+1}\rangle$ can be transformed into another proof, $\langle d_0, ..., d'_k, d_{k+1}\rangle$ say, where the contour c is added to d'_k giving d_{k+1} by commuting the applications of the add and remove contour rules. Removing c from d_{k+1} yields d'_k, showing $d_{k+1} \vdash d'_k$ and $d'_k \vdash d_{k+1}$. Hence this removal of c from d_{k+1} preserves information. Therefore $RelCF(d_{k+1}, d_0)$ is semantically equivalent to $RelCF(d_{k+1}, d_0)$ with c removed. Hence $RelCF(d_0, d_{k+1})$ is contour correctable given $RelCF(d_{k+1}, d_0)$ and we deduce that

$$EH_2(d_0, d_{k+1}) = AddContour(d_0, d_{k+1}) + RemContour(d_0, d_{k+1}) + 2 = k + 1.$$

Thus we have shown that if the relaxed remove contour rule is applied to d_0 giving d_1 then $EH_2(d_0, d_{k+1})$ provides a lower bound on proof length. No matter which rule was applied to d_0 to give d_1, it can be shown that $EH_2(d_0, d_{k+1})$ provides a lower bound on proof length. Hence $EH_2(d_1, d_2)$ is admissible.

E Clutter in Euler Diagrams

If our automatically generated proofs are to be readable then the individual diagrams must be readable. An aspect of the readability of a diagram is related to the conciseness of information representation. In first order predicate logic, for example, the sentence

$$\forall x \, \neg A(x) \qquad \text{(a)}$$

is semantically equivalent to

$$\forall x \, (\neg A(x) \wedge B(x)) \vee (\neg A(x) \wedge \neg B(x)) \qquad \text{(b)}.$$

Sentence (a) is more concise and easier to read than sentence (b). Moreover, sentence (b) could be considered *cluttered*. In [8], the amount of clutter in Euler diagrams is measured using a *clutter score*. A clutter score called the *contour score* for a diagram, d, is defined in [9] to be

$$CS(d) = \sum_{(a,b) \in Z(d)} |a|.$$

In [8] various other measure of clutter are also defined but empirical data shows that contour scoring best matches users' perceptions of clutter [9]. Contour scoring provides an indication of the conciseness of information representation: if it is possible to apply a sequence of information preserving reasoning rules to d_1 giving a diagram with a lower contour score then the information in d_1 is not represented as concisely as possible.

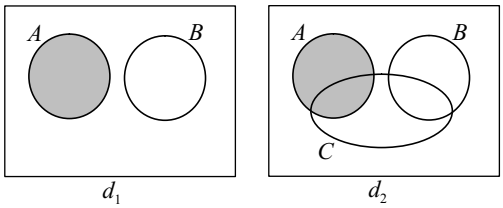

Figure 9: Clutter in Euler diagrams.

Example 2 In figure 9, d_1 and d_2 have contour scores $CS(d_1) = 2$ and $CS(d_2) = 7$. The diagrams d_1 and d_2 are semantically equivalent. In fact, d_1 can be obtained from d_2 by applying the restrictive remove contour rule. The information in d_2 can be represented more concisely, by d_1 for example.

When contours are added to a diagram, the contour score increases. In the restrictive case, for example, for any d_1 and d_2, if d_2 is obtained from d_1 by applying the add contour rule then

$$CS(d_2) = 2 \times CS(d_1) + |Z(d_1)|.$$

Both of our proof writing rules bias Edith towards proofs that contain diagrams with lower clutter scores.

F Generating Proof Tasks

In this section, we explain how the proof tasks were generated, trying to ensure randomness. Regardless of the reasoning system used, Edith can immediately check whether a proof exists, because the heuristic function takes the value of infinity whenever no proof exists (this follows from theorem 1). Hence it is not interesting, from the point of view of our analysis, to feed proof tasks to Edith in cases where there is no proof. Therefore, the proof tasks that we generate ensure that the premise semantically entails the conclusion. The premise, d_1, was randomly generated as follows.

1. Randomly choose the number, $n_contours$, of contours that will be present in d_1.

2. Generate a random set of contour labels with cardinality $n_contours$. This is $L(d_1)$.

3. The outside zone, $(\emptyset, L(d_1))$, must be in d_1 and is either shaded or non-shaded with equal probability.

4. The zones that could be in d_1, in addition to $(\emptyset, L(d_1))$, are those in $VZ(d_1) - \{(\emptyset, L(d_1))\}$. Each of the zones in $VZ(d_1) - \{(\emptyset, L(d_1))\}$ are either

   (a) in d_1 and non-shaded,

   (b) in d_1 and shaded or

   (c) not in d_1.

   Each possibility is equally likely and decided on a zone by zone basis. The set of zones deemed to be in d_1 is denoted by Z.

5. Finally, the zone set Z must be checked to see whether it satisfies the condition that there is at least one zone inside each contour. If a

contour label, l, has no zones inside then remove l from the contour label set and adjust the zones accordingly. In other words, for each $(a, b) \in Z$, replace (a, b) with $(a - \{l\}, b - \{l\})$. Repeat this process as often as needed.

Step 5 in the algorithm above ensures that each contour in d_1 contains at least one zone. Furthermore, step 5 ensures that we do not attempt to create a diagram, only to discover that it is not well-formed and have to discard it. There are many subsets of $VZ(d_1)$ that are not zone sets for any diagram. It is hoped that step 5 will save time when generating proof tasks.

To generate the conclusion, d_2, we generate a random number of contour labels and then a random set of contour labels in the same way as we did for d_1. We want d_1 to semantically entail d_2, so some zones must be present in d_2. For example, if the premise diagram is d_1 in figure 10 and d_2 is to have contour labels $L(d_2) = \{A, B, C\}$ then d_2 must contain the zones that are not shaded in d_1. As such, when generating the conclusion diagram for our

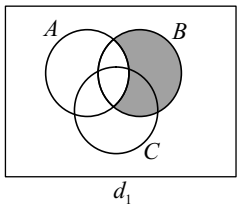

Figure 10: A randomly generated premise.

proof task, we first determine which zones must be present and non-shaded, given $L(d_2)$ and the premise diagram. The remaining zones in $VZ(d_2)$ may be present and non-shaded, present and shaded, or not present in d_2. These choices are given equal probability. Again, the resulting set of zones, Z, deemed to be present in d_2 is checked to see if it satisfies the condition that there is at least one zone inside each contour. If necessary, step 5 in the above algorithm (used to create the final zone set for d_1) is applied to Z until each contour contains at least one zone.

G Evaluation

G.1 Analysis of Time

G.1.1 Exploratory Data Analysis

Scatter plots can be seen in figure 11 that enable us to compare pairwise the times taken by Edith when using different reasoning systems. In each case,

data points below the $y = x$ line are when the reasoning system on the y axis is faster than that on the x axis and vice versa for the data points above the line.

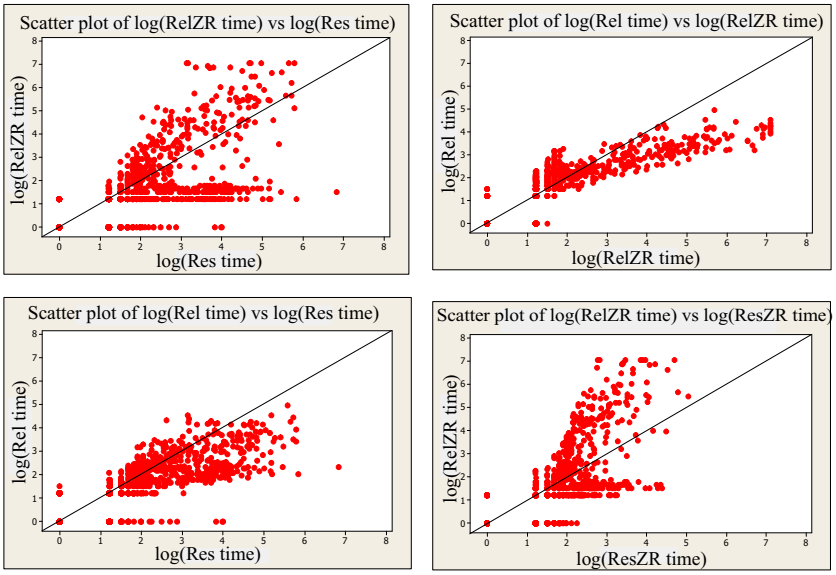

Figure 11: Scatter plots comparing time taken.

	Res Faster			No Difference			ResZR Faster		
	$N = 234$			$N = 257$			$N = 509$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	1	2	0	1	2	0	0	2
RemC	0	1	2	0	1	1	0	1	2
AddZ	0	1	2	0	0	1	1	2	2
RemZ	0	0	1	0	0	0	0	0	1
RemSh	0	0	1	0	0	1	0	1	1

Table 1: Difference measure values for Res vs Res ZR.

Summary statistics for the difference measures can be used to give an indication of the characteristics of the proof tasks for which one reasoning system is a better choice than another. Table 1 shows the median, upper and lower quartile values of the difference measures when Edith was faster using Res than ResZR and vice versa, with N being the number of times that Edith was faster using the associated system. Studying row 3, for example, the statistics suggest that Edith is likely to be faster using ResZR than Res when many zones need to be added. Tables 2 to 6 show values for the other five pairwise comparisons.

	RelZR Faster			No Difference			Res Faster		
	$N = 489$			$N = 212$			$N = 299$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	0	1	2	1	1	2
RemC	1	2	2	0	1	1	0	0	0
AddZ	1	2	3	0	0	0	0	1	2
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	1	1	1	1

Table 2: Difference measure values for Res vs RelZR.

	Rel Faster			No Difference			Res Faster		
	$N = 527$			$N = 224$			$N = 249$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	0	1	2	1	1	2
RemC	0	1	2	0	1	1	0	0	1
AddZ	1	2	3	0	0	1	0	1	1
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	1	1	1	1

Table 3: Difference measure values for Rel vs Res.

	Rel Faster			No Difference			ResZR Faster		
	$N = 458$			$N = 204$			$N = 338$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	0	1	2	0	1	2
RemC	1	2	2	0	1	1	0	0	0
AddZ	1	2	3	0	0	1	0	1	2
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	1	1	1	1

Table 4: Difference measure values for Rel vs ResZR.

G.1.2 Which is Fastest?

For each pair of reasoning systems, given a proof task we can estimate the probability that Edith will be faster with one than the other. Let $P_t(R_1, R_2, d_1, d_2)$ denote the probability that Edith will be faster at finding a proof using reasoning system R_1 than using R_2 given proof task (d_1, d_2).

	RelZR Faster			No Difference			Rel Faster		
	$N = 407$			$N = 264$			$N = 329$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	0	1	1	0	1	2
RemC	1	1	2	0	1	2	0	0	0
AddZ	1	2	3	0	0	1	0	1	2
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	1	1	1	1

Table 5: Difference measure values for RelZR vs Rel.

	RelZR Faster			No Difference			ResZR Faster		
	$N = 474$			$N = 198$			$N = 328$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	0	1	2	0	1	2
RemC	1	2	2	0	1	1	0	0	0
AddZ	1	2	3	0	0	0	0	1	2
RemZ	0	0	0	0	0	0	0	1	1
RemSh	0	0	1	0	0	1	1	1	1

Table 6: Difference measure values for RelZR vs ResZR.

We fit a log odds ratio model of the form

$$\begin{aligned}\ln\left(\frac{\hat{P}_t(R_1,R_2,d_1,d_2)}{1-\hat{P}_t(R_1,R_2,d_1,d_2)}\right) &= \beta_0 + \beta_1 AddContour(d_1,d_2)+ \\ &\quad \beta_2 RemContour(d_1,d_2)+ \\ &\quad \beta_3 AddZone(d_1,d_2)+ \\ &\quad \beta_4 RemZone(d_1,d_2)+ \\ &\quad \beta_5 RemShading(d_1,d_2)\end{aligned}$$

where $\ln\left(\frac{\hat{P}_t(R_1,R_2,d_1,d_2)}{1-\hat{P}_t(R_1,R_2,d_1,d_2)}\right)$ estimates $\ln\left(\frac{P_t(R_1,R_2,d_1,d_2)}{1-P_t(R_1,R_2,d_1,d_2)}\right)$. If $\beta_i = 0$ then the associated measure is not significant. For the Rel and ResZR system the fitted model is

$$\begin{aligned}\ln\left(\frac{\hat{P}_t(Rel,ResZR,d_1,d_2)}{1-\hat{P}_t(Rel,ResZR,d_1,d_2)}\right) &= -2.6377 \\ &\quad +0.3604 AddContour(d_1,d_2) \\ &\quad +1.3870 RemContour(d_1,d_2) \\ &\quad +0.5506 AddZone(d_1,d_2) \\ &\quad +0.3179 RemZone(d_1,d_2)\end{aligned}$$

Table 7 shows some estimated probabilities (to 2 d.p.) given various values of the difference measures (because the remove shading measure is not included

in the model, it has no effect on our estimated probability, so is not in the table) which we now discuss. Suppose we have a proof task, (d_1, d_2),

Measures				
AddC	RemC	AddZ	RemZ	$\hat{P}_t(Rel, ResZR, d_1, d_2)$
0	4	4	0	0.99
0	4	0	0	0.95
4	0	0	0	0.23
4	0	0	6	0.67
0	0	2	3	0.36
0	0	4	4	0.70

Table 7: Estimated probabilities.

where we need to remove four contours from d_1 to obtain d_2, so $RemC = RemContour(d_1, d_2) = 4$. It is likely that, using ResZR, some zones will need to be added before contours can be removed, so the add zone measure, AddZ, is unlikely to be zero. In such cases, we might expect Edith to perform better using Rel than ResZR because it can remove the contours without first adding the zones. Row 1 of table 7 shows that this is, indeed, likely to be the case. Reducing the value of AddZ does not have much impact on the probability, see row 2 for an example.

If all we have to do is add contours to d_1 in order to obtain d_2 then we might expect Edith to perform no worse using ResZR than Rel because, unlike the restrictive add contour rule, there are potentially many ways of adding a contour using the relaxed add contour rule; row 3 shows an example situation providing evidence to support this intuition. However, once we need to start removing zones, too, we might expect Rel to perform better than ResZR: the relaxed add contour rule can be applied in such a way that it is equivalent to an application of the restrictive add contour rule followed by subsequent applications of the remove zone rule; row 4 provides evidence to support this intuition.

If there are zones to be added and zones to be removed from d_1 then the highly explosive nature of the add region and remove region rules (in the sense that as the number of missing zones and shaded zones increases, the number of ways in which these rules can be applied explodes) may mean that ResZR is a better choice. Row 5 shows that in relatively simple cases, ResZR is likely to be no worse a choice than Rel, whereas row 6 shows that when we detect many zone differences, the explosive nature of the region rules is likely to outweigh the cost of producing longer search trees (that is, Rel is likely to be a better choice than ResZR).

As a way of evaluating our model, we calculated the probability of Rel being faster than ResZR using a new data set of 500 proof tasks with cut-off time of 10 minutes; this cut-off time was deemed sufficient because most proof tasks were solved very quickly in our 1000 data set. We plotted these probabilities against the time differences, where the times are in milliseconds; this is the lefthand plot in figure 12 whereas the righthand plot restricts the y-axis to between ±300 milliseconds. The plots illustrate that the prediction of the best reasoning system to use is usually correct; most points with probability of less than 0.5 are above the x-axis (i.e. ResZR was faster) and most points with probability of more than 0.5 are below the x-axis (i.e. Rel was faster). Indeed, typically the bigger the time difference the further away from 0.5 the probability becomes.

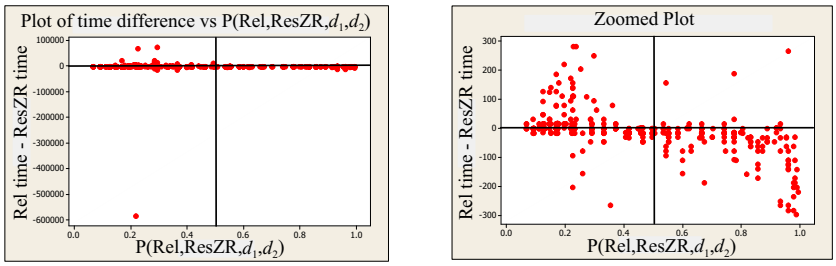

Figure 12: Plots showing time differences against probabilities.

For the remaining five log odds ratio models, table 8 summarizes the coefficients (β_i's) of the restrictive difference measures with a dash indicating that a variable is not significant (and, therefore, not included in the model).

	β_0	AddC	RemC	AddZ	RemZ	RemSh
(ResZR, Res)	-1.8561	0.2293	–	0.9104	0.7745	0.4366
(RelZR, Res)	-2.7737	0.2819	1.5874	0.8389	0.2663	–
(Rel, Res)	-2.5726	0.2868	0.3438	1.0553	0.7226	–
(RelZR, Rel)	-2.7340	0.3219	1.0726	0.5300	–	0.5818
(RelZR, ResZR)	-3.0138	0.2827	2.0047	0.6475	0.2768	–

Table 8: Coefficients for the log odds models.

These models can be used to estimate the probability that Edith will be faster using one reasoning system than another given the measure values. The coefficients of the difference measures are all positive. It follows that the odds of R_1 being faster than R_2 increase as the values of the difference measures

increase. From this we deduce that $\hat{P}_t(R_1, R_2, d_1, d_2)$ increases as the values of the difference measures increase (although the increased $\hat{P}_t(R_1, R_2, d_1, d_2)$ could be less than $\frac{1}{2}$; if $\hat{P}_t(R_1, R_2, d_1, d_2) < \frac{1}{2}$ then a substantial increase in the measure values may be required before it is greater than $\frac{1}{2}$); that is, the probability that Edith is faster using R_1 than R_2 increases as the complexity of the proof task increases.

Each of the models above has a negative constant term; from this, it follows that the odds favour R_2 when the measure values are small; equivalently, $\hat{P}_t(R_1, R_2, d_1, d_2) < \frac{1}{2}$. Indeed, in the cases where the coefficient for a measure is small, the measure value may need to be large for the probability that Edith is faster using R_1 than R_2 to be greater than $\frac{1}{2}$.

Plots showing time difference against probability, produced using the new data set containing 500 proof tasks, can be seen in figures 13 to 17.

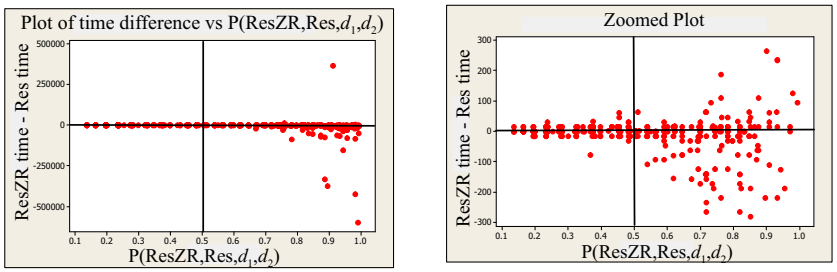

Figure 13: Plots showing time differences against probabilities for ResZR and Res.

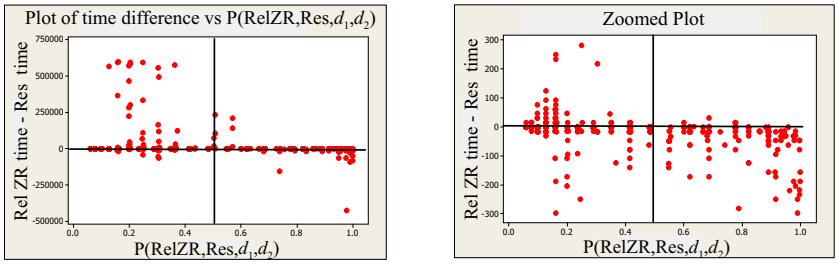

Figure 14: Plots showing time differences against probabilities for RelZR and Res.

G.2 Analysis of Space

As a measure of space efficiency, we can use the size of the search tree generated by Edith; we counted the number of nodes in the tree (that is, the number of diagrams created when attempting to solve a proof task).

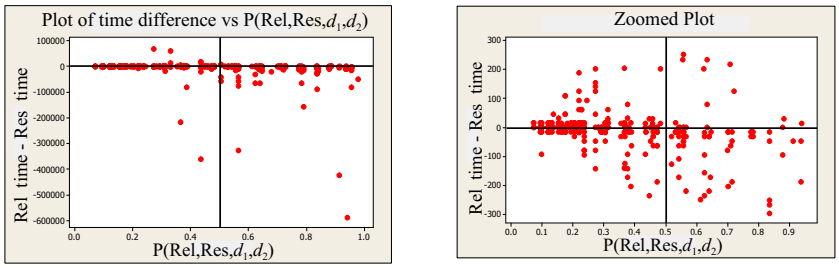

Figure 15: Plots showing time differences against probabilities for Rel and Res.

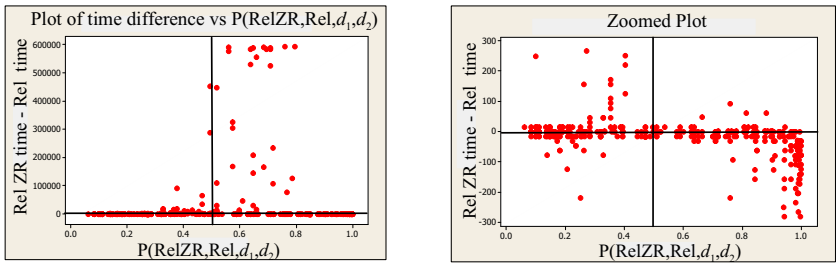

Figure 16: Plots showing time differences against probabilities for RelZR and Rel.

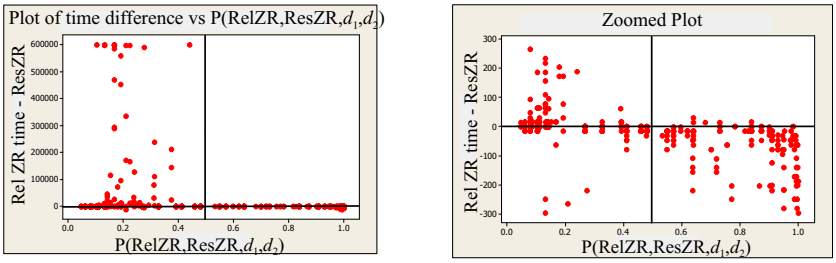

Figure 17: Plots showing time differences against probabilities for RelZR and ResZR.

G.2.1 Exploratory Data Analysis

Scatter plots can be seen in figure 18 that enable us to compare pairwise the sizes of the search trees generated by Edith when using different reasoning systems. We have plotted the logs of the sizes of the search trees, $\log(Reasoning\ system\ STS)$. In each case, data points below the $y = x$ line are when the reasoning system on the y axis has a smaller associated search tree than that on the x axis and vice versa for the data points above the line.

Summary statistics for the difference measures can be used to give an indication of the characteristics of the proof tasks for which one reasoning

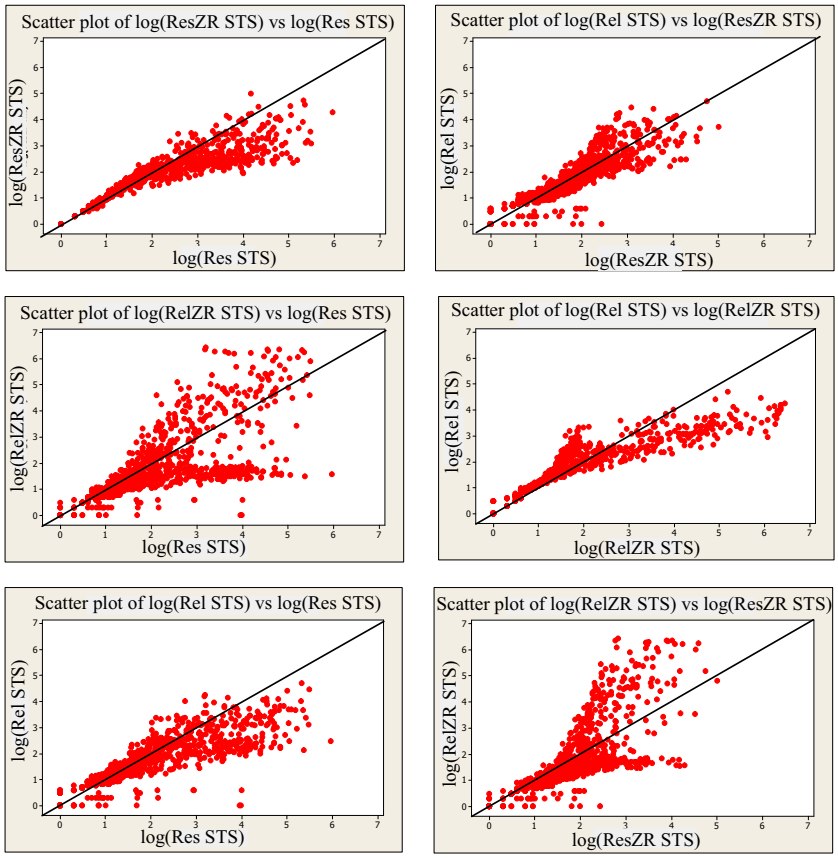

Figure 18: Scatter plots comparing the sizes of the search trees.

	Res Smaller			No Difference			ResZR Smaller		
	$N = 328$			$N = 198$			$N = 474$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	1	2	1	1	2	0	0	1
RemC	0	1	2	0	1	1	0	1	2
AddZ	0	1	2	0	0	0	1	2	3
RemZ	0	0	1	0	0	0	0	0	1
RemSh	0	1	1	0	0	0	0	1	1

Table 9: Difference measure values for Res vs ResZR.

system is a better choice than another in terms of the size of the search tree. These can be seen in tables 9 to 14.

Table 15 summarizes the descriptive statistics for the number of vertices in the search tree created when finding a shortest proof using each reasoning system. For example, using Res the search tree contained between 1 vertex

	RelZR Smaller			No Difference			Res Smaller		
	$N = 553$			$N = 153$			$N = 294$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	1	2	3	0	1	2
RemC	1	1	2	0	0	1	0	0	0
AddZ	1	2	2	0	0	0	0	1	2
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	0	1	1	1

Table 10: Difference measure values for RelZR vs Res.

	Rel Smaller			No Difference			Res Smaller		
	$N = 541$			$N = 34$			$N = 425$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	1	1	1	1	1	2
RemC	0	1	2	0	0	1	0	0	1
AddZ	1	2	3	0	0	1	0	0	1
RemZ	0	0	1	0	0	0	0	0	1
RemSh	0	0	1	0	0	0	0	1	1

Table 11: Difference measure values for Rel vs Res.

	Rel Smaller			No Difference			ResZR Smaller		
	$N = 477$			$N = 36$			$N = 487$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	1	1	1	0	1	2
RemC	1	1	2	0	0	1	0	0	1
AddZ	1	2	2	0	0	1	0	0	2
RemZ	0	0	1	0	0	0	0	0	1
RemSh	0	0	1	0	0	0	0	1	1

Table 12: Difference measure values for Rel vs ResZR.

and nearly 1 million vertices, with the middle half of the proof tasks being solved when creating between 13 and 771 vertices. The Q3 column shows that $\frac{3}{4}$ of the search trees contained fewer than 800 vertices, regardless of the reasoning system used. As well as taking the shortest time in most cases, using RelZR Edith usually created smaller search trees than with the other systems (Q3 is smallest for RelZR), but was the only system with which Edith created search trees containing more than 1 million vertices. Edith can also

	RelZR Smaller			No Difference			Rel Smaller		
	$N = 694$			$N = 58$			$N = 248$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	1	1	0	0	1	0	1	2
RemC	0	1	2	0	1	1	0	0	0
AddZ	0	1	2	0	0	1	0	1	2
RemZ	0	0	0	0	0	0	1	1	2
RemSh	0	0	1	0	0	0	1	1	1

Table 13: Difference measure values for RelZR vs Rel.

	RelZR Smaller			No Difference			ResZR Smaller		
	$N = 542$			$N = 167$			$N = 291$		
Measure	Q1	Median	Q3	Q1	Median	Q3	Q1	Median	Q3
AddC	0	0	1	1	2	3	0	1	2
RemC	1	2	2	0	0	1	0	0	0
AddZ	1	2	2	0	0	0	0	1	2
RemZ	0	0	1	0	0	0	0	1	1
RemSh	0	0	1	0	0	0	1	1	1

Table 14: Difference measure values for RelZR vs ResZR.

	Min	Q1	Median	Q3	Max
Res	1	13	63	771	902969
ResZR	1	14	65	233	99956
RelZR	1	9	29	114	2759995
Rel	1	14	47	231	51045

Table 15: Descriptive statistics for space used.

create very large search trees using Res, seen by its maximum search tree size, whereas Edith never created a search tree containing more than 100,000 vertices when using ResZR and Rel.

G.2.2 Which Has The Smallest Search Tree?

Given a proof task, (d_1, d_2), we would like to know whether Edith will create a smaller search tree using Rel than when using ResZR. Let $P_s(Rel, ResZR, d_1, d_2)$ denote the probability that Edith will create a smaller search tree using reasoning system Rel than using ResZR given (d_1, d_2). We fitted a log odds

ratio model, where $\hat{P}_s(Rel, ResZR, d_1, d_2)$ estimates $P_s(Rel, ResZR, d_1, d_2)$:

$$\begin{aligned}\ln\left(\frac{\hat{P}_s(Rel,ResZR,d_1,d_2)}{1-\hat{P}_s(Rel,ResZR,d_1,d_2)}\right) &= -2.2308 \\ &\quad +0.8901 RemContour(d_1,d_2) \\ &\quad +0.8349 AddZone(d_1,d_2) \\ &\quad +1.2143 RemZone(d_1,d_2) \\ &\quad -0.7925 AddZone(d_1,d_2) \times RemZone(d_1,d_2) \\ &\quad +0.7362 RemContour(d_1,d_2) \times RemZone(d_1,d_2)\end{aligned}$$

(the add contour measure was not significant, but there were three significant interactions included in the model) which had goodness of fit measure 1.03.

Since the coefficients of the measure values are not all of the same sign, we cannot interpret the model as easily as in the time taken case. Table 16 shows some estimated probabilities (to 2 d.p.) given various values of the difference measures which we now discuss.

Measures			
AddZ	RemZ	$\hat{P}_s(Rel, ResZR, d_1, d_2)$	$\hat{P}_t(Rel, ResZR, d_1, d_2)$
1	0	0.09	0.11
2	4	0.17	0.43
3	4	0.08	0.57
4	4	0.04	0.70

Table 16: Estimated probabilities.

Suppose we have a proof task, (d_1, d_2), where the add zone measure is 1 and other measures are all zero (row 1 of the table); all we need to do is add zones. Then we would expect ResZR to be no worse a choice in terms of space and time (see row 1). If we increase both the add zone and remove zone measure values then, in terms of the size of the search tree, ResZR remains the most likely best choice in terms of space (see rows 2 – 4). However, although initially Edith is much more likely to be faster using ResZR than Rel, as these two measures increase this fails to be the case, with Rel becoming faster; our notion of best choice differs between time and the size of the search tree.

The scatter plots in figure 19, created using the 500 data set show that the model is reasonably good.

For the remaining five log odds ratio models, table 17 summarizes the coefficients (β_i's) of the restrictive difference measures with a dash indicating that a variable is not significant (and, therefore, not included in the model).

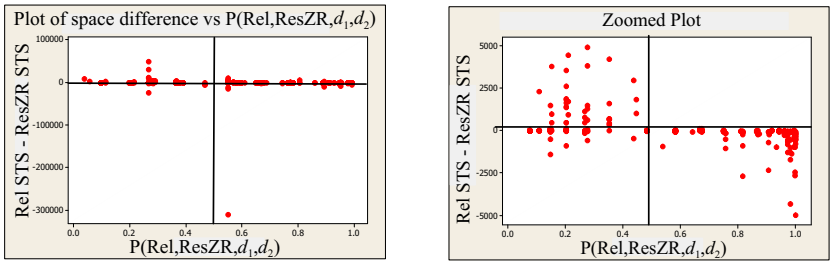

Figure 19: Plots showing size of search tree differences against probabilities.

	β_0	AddC	RemC	AddZ	RemZ	RemSh
(ResZR, Res)	-2.0342	–	–	0.9063	0.7584	0.8798
(RelZR, Res)	-1.8600	-0.2512	1.9806	0.4178	0.6818	–
(Rel, Res)	-1.9721	–	0.6842	1.1281	1.1182	-0.4953
(RelZR, Rel)	0.5546	–	1.8902	-0.4292	-0.7161	–
(RelZR, ResZR)	-2.5406	–	2.4667	0.4001	0.7613	–

Table 17: Coefficients for the log odds models.

These models can be used to estimate the probability that Edith will create a smaller search tree using one reasoning systems than another.

Plots showing differences between the sizes of the search trees against probability, produced using a new data set containing 500 proof tasks, can be seen in figures 20 to 24.

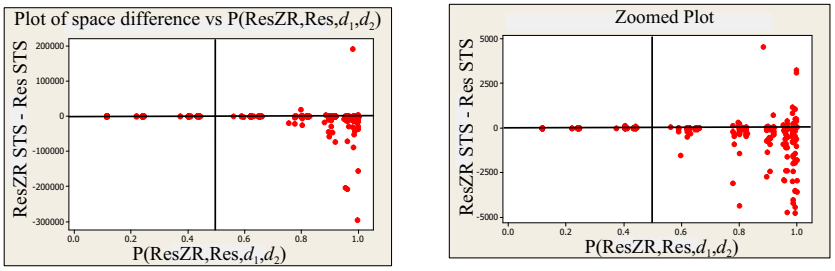

Figure 20: Plots showing size of search tree differences against probabilities for ResZR and Res.

G.3 Using Heuristics

In order to establish that our heuristics bring benefits over a breadth first search, we generated a random sample of 500 proof tasks. For each task and

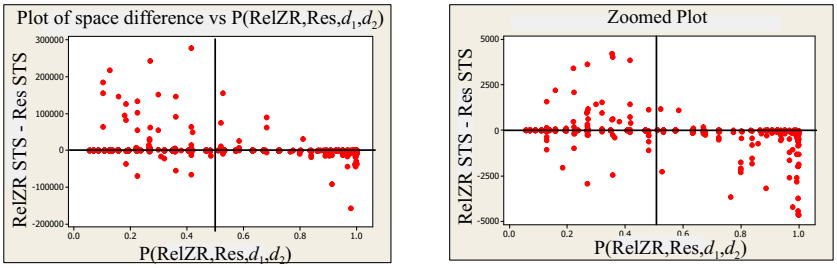

Figure 21: Plots showing size of search tree differences against probabilities for RelZR and Res.

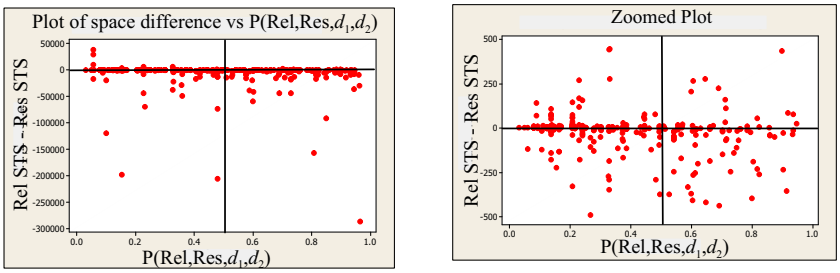

Figure 22: Plots showing size of search tree differences against probabilities for Rel and Res.

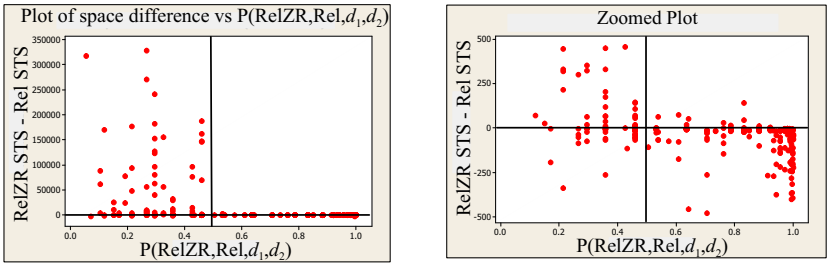

Figure 23: Plots showing size of search tree differences against probabilities for RelZR and Rel.

each rule set, we asked Edith to find a proof using the heuristic and a proof using a breadth first search. However, we decided to stop Edith searching whenever no proof had been found after ten minutes; this cut-off time was deemed sufficient since most proofs in the 1000 sample were found much more quickly than this.

Table 18 shows the descriptive statistics for each reasoning system where we have computed the time differences: time (in milliseconds) taken with a breadth first search minus time taken using the heuristic. Min is the minimum time difference, Q1 is the 25th percentile, Q3 is the 75th percentile

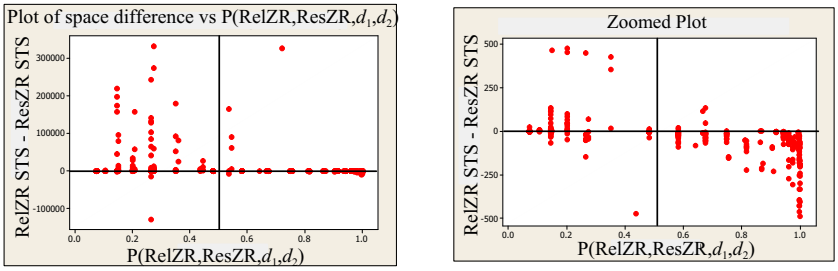

Figure 24: Plots showing size of search tree differences against probabilities for RelZR and ResZR.

and Max is the maximum time difference. For example, using Res, Edith was mostly faster using the heuristic than when conducting a breadth first search, with the median time difference being 235 milliseconds. The maximum time differences are all very high with Res being 599922, for example. The maximum differences all occurred when Edith failed to find a proof using a breadth first search. With the cut-off time of 10 minutes being 600000 milliseconds, these maximum time differences illustrate that sometimes a proof was found almost instantly using the heuristic but was not found when conducting a breadth first search; as an illustration, $599922 = 600000 - 78$, so a proof was found in 78 milliseconds using Res with the heuristic but timed out using a breadth first search. Clearly, this indicates that using a heuristic can bring massive time savings. Table 19 shows similar values for the differences between the sizes of the search trees generated, where the size is the number of vertices. The three systems that have negative minimums virtually always had smaller search trees when using the heuristic; Res had a negative difference twice, ResZR once and RelZR three times.

	Min	Q1	Median	Q3	Max
Res	-46	0	235	226703	599922
ResZR	-31	0	243	291754	599890
RelZR	-234	0	15	531	599062
Rel	-172	0	16	7089	599907

Table 18: Descriptive statistics for the time differences.

Acknowledgements Andrew Fish and Gem Stapleton were partially supported by EPSRC grants GR/R63516 and GR/R63509 for the Reasoning with Diagrams project and EP/E011160 for the Visualization with Euler Diagrams project. Gem Stapleton is supported by a Leverhulme Trust Early

	Min	Q1	Median	Q3	Max
Res	-35501	9	1170	139994	304954
ResZR	-72488	10	1535	235625	391200
RelZR	-63792	0	165	4519	485106
Rel	0	0	235	20465	689868

Table 19: Descriptive statistics for the search tree size differences.

Career Fellowship. We thank John Taylor and John Howse for helpful comments on an earlier draft of this paper.

References

[1] Alexoudi, M., C. Zinn, and A. Bundy. English Summaries of Mathematical Proofs. In *Workshop on Computer-supported mathematical theory development at the International Joint Conference on Automated Reasoning*, pages 49–60, 2004.

[2] Bundy, A. A Very Mathematical Dilemma. In *Computer Journal*, 49(4):480–486, 2006.

[3] Dechter, R. and J. Pearl. Generalized Best-First Search Strategies and the Optimality of A^*. *Journal of the Association for Computing Machinery*, 32(3):505–536, 1985.

[4] Flower, J. and J. Howse. Generating Euler Diagrams. In *International Conference on the Theory and Application of Diagrams*, LNAI 2317, pages 61–75, Georgia, USA, April 2002. Springer-Verlag.

[5] Fomin, F., D. Kratsch, and G. Woeginger. Exact (Exponential) Algorithms for the Dominating Set Problem. In *Proceedings of the 30th Workshop on Graph Theoretic Concepts in Computer Science*, LNCS 3353, pages 245–256. Springer-Verlag, 2004.

[6] Howse, J., G. Stapleton, J. Flower, and J. Taylor. Corresponding Regions in Euler Diagrams. In *International Conference on the Theory and Application of Diagrams*, LNAI 2317, pages 146–160, Georgia, USA, April 2002b. Springer-Verlag.

[7] Howse, J., G. Stapleton, and J. Taylor. Spider Diagrams. *LMS J. Computation and Mathematics*, 8:145–194, 2005b.

[8] John, C. Measuring and Reducing Clutter in Euler Diagrams. In *Euler Diagrams Workshop*, vol. 134 *ENTCS*, pages 103–126, Brighton, Elsevier, 2005.

[9] John, C. and A. Fish and J. Howse and J. Taylor Exploring the Notion of Clutter in Euler Diagrams. In *Proceedings of 4th International Conference on the Theory and Application of Diagrams*, Springer, to appear June 2006.

[10] Rodgers, P., P. Mutton, and J. Flower. Dynamic Euler Diagram Drawing. In *IEEE Symposium on Visual Languages and Human Centric Computing*, pages 147–156, Rome, September 2004. IEEE Computer Society Press.

[11] Stapleton, G. J. Masthoff, J. Flower, A. Fish and J. Southern. Automated Theorem Proving in Euler Diagram Systems. Submitted to Journal of Automated Reasoning, 2006.

[12] Swoboda, N. *Implementing Euler/Venn Reasoning Systems*, In M. Anderson, B. Meyer and P. Oliver, editors, Diagrammatic Representation and Reasoning, pages 371–386. Springer-Verlag, 2001.