Explanation-Based Learning With Analogy for Impasse Resolution

Matt Timperley[a,*], Maizura Mokhtar[b], Gareth Bellaby[c], Joe Howe[d]

[a] *UCLAN Energy, University of Central Lancashire, UK*
[b] *Rolls-Royce University Technology Centre, University of Sheffield, UK*
[c] *School of Computing, Engineering and Physical Sciences, University of Central Lancashire, UK*
[d] *Thornton Energy Institute, University of Chester, UK*

Abstract

This paper proposes an algorithm for the inclusion of analogy into Explanation-Based Learning (EBL). Analogy can be used when an impasse is reached to extend the deductive closure of EBL's domain theory. This enables the generation of control laws, via EBL, for hardware which is not catered for in the domain theory. This advantage addresses a problem which represents a dearth in the current literature. Integrated Modular Avionics (IMA) literature has thus far been concerned with the architectural considerations. This paper seeks to address the impact of hardware changes on the controllers within an IMA architecture. An algorithm is proposed and applied to control an aviation platform with an incomplete domain theory. Control rules are generated when no deductive explanations are possible, which still reflect the intent of the domain theory.

Keywords: Explanation-Based Learning, Impasse Resolution, Analogy

1. Introduction

Modularity in aviation has been gaining interest (Wilson & Preyssler, 2009) because of both the additional future-proofing inherent in having easily replaceable modules and the flexibility of role that this engenders (Committee on Materials, Structures, and Aeronautics for Advanced Uninhabited Air Vehicles, Commission on Engineering and Technical Systems, 2000). Modularity makes changing the hardware of platforms easier in order to fit specific missions, as advocated in (Lopez et al., 2008). Changes to a hardware platform may require changes to the control software. The impact of changes in the constituent

*Corresponding author
Email addresses: MTimperley@uclan.ac.uk (Matt Timperley), M.Mokhtar@sheffield.ac.uk (Maizura Mokhtar), GJBellaby@uclan.ac.uk (Gareth Bellaby), J.Howe@chester.ac.uk (Joe Howe)

hardware of a platform on the software control systems, which operate on said platform, forms a gap in the existing literature, as do coping strategies for this scenario.

Modular Avionics are aviation electrical systems which are integrated using a modular design paradigm. The components forming a modular aviation platform can be easily swapped (Faculty & Kahn, 2001). An Integrated Modular Avionics (IMA) architecture involves the interconnection of physically separate hardware components which share computing hardware, whilst remaining logically separate (Lopez et al., 2008).

The interest in modularity in aviation hardware, and the current gap in the literature highlights an open problem. It would be advantageous for software to adapt to control new items of hardware with a minimum of extra work. This objective can be satisfied in this work by the generation of control rules for an item of hardware which lies outside the original deductive closure of the domain theory.

The foundations of generating fuzzy rules from Explanation-Based Learning (EBL) explanation structures are in (Timperley, 2015). An assumption employed was that an appropriate, general, domain theory had been elicited. In this paper it is assumed that the domain theory is incomplete for the intended purpose. This new assumption leads to an impasse being reached when attempting to form an explanation. As explanations are used as the basis of new fuzzy rules, no control laws will be derived.

EBL is focussed on generalisation, usually through the introduction of variables. This facet of EBL is what lends itself to modular control. In this work EBL can be further extended to support the generalisation of predicates, using analogies as supporting evidence.

The aim of this paper is to incorporate analogical learning into Explanation-Based learning (EBL). Analogy is proposed as a way to increase the deductive closure of the domain theory upon reaching an impasse. This allows a controller to generate rules for hardware outside of its original design, and therefore not mentioned in its domain theory. Such a situation can occur when the hardware of the controlled platform changes.

1.1. Analogy

A previous work has augmented EBL to reach a deductive solution by combining two incomplete domain theories (Hirowatari & Arikawa, 1994). Analogy is used as the basis for linking predicates within two incomplete domain theories. The combination of domain theories leads to a deductive solution, in contrast to what is proposed in this work. Augmentation of EBL with analogical capabilities is an area which merits further work. This paper builds upon the previous work by allowing analogy to extend an existing domain theory.

Analogy can be employed to continue explanation upon reaching an impasse. It is likely that a new component will share both similarities to other components and an energy management strategy. This transfer of knowledge between situations can be achieved by derivational analogy (Carbonell, 1983). Analogy has been used to map different situations to one another, in order to apply prior knowledge to a new situation (Huhns & Acosta, 1988); thereby performing transfer learning.

This paper aims to use EBL and analogy in order to generate useful control rules for hardware which is outside the design of the controller. The objectives of this paper are to: (i) Introduce an algorithm which augments EBL with analogy to perform transfer learning; and (ii) demonstrate the the augmented EBL algorithm can generate rules which are useful but lie outside the deductive closure of the domain theory. The objectives will be demonstrated by applying the augmented EBL algorithm to control a hardware platform where no rules can be generated deductively i.e. without transfer learning. If the generated rules control the platform in a manner which reflects the intent of the domain theory then the second objective will be satisfied.

This paper is further divided into 5 sections. Section 2 compares the combination of EBL and analogy presented in this paper to existing methods. The advantages of combining EBL and analogy are described in Section 3, which also provides the proposed algorithm and satisfies the first objective of this paper. In order to satisfy the second objective of this paper the algorithm is applied to pitch control. The experiment described in Section 4 uses a domain theory for throttle control which cannot produce rules for pitch deductively. Analogy is used to generate rules for pitch. The result of applying these rules is given in Section 5. The paper concludes with Section 6.

2. Related Work

Transfer learning can be applied as an alternative to an impasse, such as in (Burstein, 1986). In other applications of analogical reasoning (e.g. (Carbonell, 1983), (Klenk & Forbus, 2009)) a known solution to a similar problem is found and mutated to be applicable to the current problem.

Case Based Reasoning (CBR) was the method of analogical reasoning in (Carbonell, 1983) and (Klenk & Forbus, 2009). CBR uses similarities between previous solutions to various problems to solve the one at hand, the target (de Mántaras & Plaza, 1997). Closely similar solutions are modified to solve the target problem. Similarity metrics are used to guide searches for similar solutions. In this work a similarity between terms is used as the basis for analogy.

If comparing the technique proposed with that of derivational analogy (Carbonell, 1983), the analogue is a concept rather than a situation. Another difference is, the presented technique, rather than tailoring a previous solution to a new situation, generalises a previous line of reasoning within a solution, to more concepts. Applying analogy to only a single line within an example is similar to when students refer to a specific line of a previous example to justify some reasoning rather than the example as a whole (VanLehn & Jones, 1998). This technique also shares some conceptual similarities with (Ishikawa & Terano, 1996).

Both this work and that of Könic et al (Könik et al., 2009) map between concepts in order to apply knowledge from one situation to another. There are some differences between this work and (Könik et al., 2009). This technique is proposed to derive links between previously unrelated concepts. These links are embodied in a new, more abstract, concept definition. This technique has a different method of application, as an alternative to failure. Both works use a goal and explanation, as a bias, to constrain the possible analogies. This technique also maps between potentially overlapping concepts rather than like terms.

This type of analogy, from information in (Falkenhainer, 1987), could be stated as using similarity-based generalisation in order to perform a form of analogical reasoning. The analogical reasoning is restricted to generalising information to hold to more concepts than when derived. The algorithm forms a new concept from two existing definitions, keeping the parts specific to both and introducing variables where variation exists. In this regard the algorithm is similar to EGGS (Mooney et al., 1986) which performs generalisation by applying the aggregation of substitutions within an explanation to the whole explanation.

In this work an analogy is seen as evidence for predicate generalisation or swapping. The generalisation can either replace the target predicate or be taken as evidence for swapping the source for the target. Analogies could be generated using different techniques in order to generate evidence for predicate swapping. One such method which is capable of deriving analogies from natural language is (Cambria et al., 2015). It may be possible to use such techniques and either apply them to domain theories. However, if domain theories are lacking in data richness then the system in (Cambria et al., 2015) could be applied to the expert knowledge from which the domain theory was designed.

The algorithm proposed differs from (Hirowatari & Arikawa, 1994) by increasing slightly the deductive closure of the program in order to apply analogical reasoning in more restricted form. This increase in deductive closure is based on the assumption that the addition of a new concept increases the number of facts that can be deduced within the system.

An application of analogical learning with EBL (Hirowatari & Arikawa, 1994) was able to construct explanations from incomplete domain theories where the combination of domains made a deductive explanation possible. This work proposes that less exact analogues can be used to derive relations between terms with different predicates. Rather than establishing a link between the same terms in different domains, the emphasis is on relating different terms within the same domain. In this paper the algorithm proposed goes further and extends a single incomplete domain theory using internal similarities between two predicates. This is achieved by deriving an abstraction which encompasses both predicates.

3. Abstraction Derivation

This paper proposes the derivation of an analogical link between two deductively separate concepts. This link is formed by the generation of a new concept, which requires commonalities between the two target concepts to be satisfied. The new concept is an abstraction of the two target concepts. Membership of this abstraction implies that concepts are analogous in the manner given by the abstract concept definition. This could be thought of as weakening the preconditions on the target concepts to form an abstraction.

An advantage of EBL can be exploited during the definition of an analogical similarity; training examples encountered can provide an inductive bias for disregarding terms that are not relevant.

During explanation an impasse may be reached when no deductive option remains, but an inductive leap is possible. Inductive leaps are realised by replacing one concept in a rule with another. This gives rise to a new rule which relates previously unrelated concepts, linked by analogy. These analogous concepts are not deductively entailed. The difference in predicates prohibits their mutual unification to explain the same goal. Additions to the domain theory which embody the similarities between the two concepts are used as a basis for replacing one predicate with another.

Rather than generalising a rule by abstracting specific variable bindings from terms, as in EBL (Ellman, 1989), two concept definitions are linked by a more abstract class so that the terms themselves can be replaced. This is similar to assuming that the predicates themselves could be bound differently. The abstract class can be formed from commonalities between otherwise disparate concept definitions. This class embodies an analogy and is expressed as a new rule. The algorithm will now be presented.

3.1. Forming an Abstract Definition

The two concepts which are used to form an abstraction are the source and the target. The source is the definition of the predicate which causes an impasse. The target is a concept searched for, using the parameters of the source as a search bias.

Mapping establishes the similarities between two identified concepts, taken from the model of analogy presented in (Gentner, 1983). An approach to achieve this within EBL is proposed. The approach for forming a new, more abstract, definition from two target concept definitions is:

1. Collect like terms in the leaf nodes of both concepts and reduce the number of occurrences to one by introducing variables and forming a substitution. Do this for both, or all, concepts being considered.
2. Discard any terms that do not appear in all concept definitions. When considering many concept definitions, one may discard from consideration concepts that lack common terms rather than the terms themselves. As long as a relationship between at least two concepts is established, this is permissible.
3. Use these common terms to form the precedents for a new concept definition. This definition captures some similarity between the concepts considered which is assumed to be absent from the initial domain theory.

This approach has been formalised in Algorithms 1 and 2. Algorithm 1 selects the analogy with the most terms in it, corresponding to matches between any combination of definitions for both source and target terms. This representation calculates all the possible abstractions between each definition of both concepts, sorted in descending order of complexity. In practice, the most complex abstraction is taken and assumed to show a closer connection between two concepts. Algorithm 2 forms the abstraction between two concepts.

Algorithm 1 Impasse(term source)

```
Require: source is a ground atom
  targets ← rules containing predicate(s) with arguments matching the source.

  SourceDefs ← rules previously derived that imply the source, stored within
  the rule-base.
  // Collect analogies between each definition of the source and target. These
  analogies are stored in "analogies" which is sorted by number of terms (descending) in the abstraction.
  for all target in targets do
    targetDefs ← rules previously derived that imply the source, stored within
    the rule-base.
    for all targetDef in targetDefs do
      for all sourceDef in sourceDefs do
        Abstract(source, target)
        store target in analogies along with its abstraction (targetDef).
      end for
    end for
  end for
  add the target, topmost from analogies, under source in the explanation structure.
```

Terms are discarded which are not common to both source and target definitions, as these are assumed to be irrelevant to whatever similarity is being captured. This is to facilitate the capture of analogues that have the most in common. More general analogues that make a weaker claim of similarity between concepts could also be useful but could more readily lead to over-generalisation. Also, it may be computationally wasteful to derive very general analogues since these could potentially be applied often to little effect.

Over-generalisation and over-specialisation have been the topic of previous works such as the EGGS algorithm (Mooney et al., 1986). The generalisation used to form the abstraction is similar to the EGGS algorithm. However, there are some differences: it is applied across two explanations, like predicates are aggregated within a single explanation (source or target definitions), and the generalised terms are added to a new explanation rather as replacements to an existing explanation. This algorithm will be illustrated using an example.

3.2. An Example

The analogical EBL algorithm can be applied to an energy management example. Consider a battery powered system being altered to be powered by a

Algorithm 2 Abstract(term source, term target)

```
Require: source and target predicates are ordered.
  get the first terms from source and target concepts.
  for all predicates in source do
    if current predicates from source and target concepts differ then
      get next source predicate.
    else
      while the next predicate in the source matches the current one do
        note the arguments, by arity, from source term.
        get the next term from source.
      end while
      while the next predicate in the target matches the current one do
        note the arguments, by arity, from target.
        get the next term from target.
      end while
      for all arguments from source for this predicate. do
        if there is a corresponding argument in target then
          add the current predicate, with the matching argument, to the abstraction.
        else
          // this should be limited to once per predicate
          add the current predicate, with an introduced variable as the argument, to the abstraction.
        end if
      end for
    end if
  end for
  return the abstraction.
```

fuel cell. The domain theory and goal are given in Table 1.

Table 1: A sample input for EBL

Goal
can_supply(fuel_cell_1, ASI_demand)
Training Data
max_output(fuel_cell_1, 50)
demand(ASI, ASI_demand)
Domain Theory
limited(X, charge) + waste(X, heat) + requires(X, charge) + produces(X, energy) → battery(X)
fuel(X, hydrogen) + waste(X, heat) + waste(X, water) + requires(X, hydrogen) + produces(X, energy) → fuel_cell(X)
battery(X) + max_output(X,O) > demand(C,Y) → can_supply(X,C)
fuel_cell(fuel_cell_1)
battery(battery_1)

The EBL algorithm reaches an impasse when attempting to explain the goal: *battery(fuel_ cell_ one)*. This is shown in Figure 1. The boxed area in Figure 1 is the point an impasse is reached. Single lines denote implications. The double line is where the algorithm generates an abstraction to link the source and target concepts.

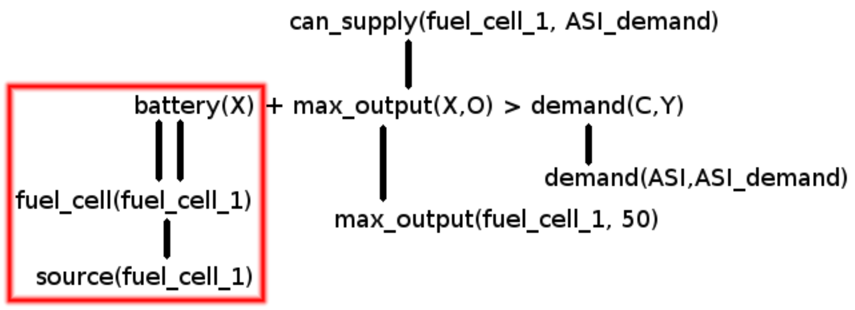
can_supply(fuel_cell_1, ASI_demand)
battery(X) + max_output(X,O) > demand(C,Y)
fuel_cell(fuel_cell_1)
demand(ASI,ASI_demand)
max_output(fuel_cell_1, 50)
source(fuel_cell_1)

Figure 1: Example explanation structure starting with battery oriented rules and using abstraction to substitute battery for *source*. Single lines depict implications and can be read that the lower line implies the statement above. A box shows where the proposed technique is being applied with the double line representing an abstract similarity.

The algorithm uses the unbound goal which caused an impasse as the source concept. In this case: *battery(X)*. Target concepts are identified as statements taking the form *X(fuel_ cell_ 1)*: In this case, *fuel_ cell(fuel_ cell_ 1)*.

The rules which imply source and target concepts are taken as their definitions. In this case the source definition is: *limited(X, charge) + waste(X, heat) + requires(X, charge) + produces(X, energy) → battery(X)*. The target definition is: *fuel(X, hydrogen) + waste(X, heat) + waste(X, water) + requires(X,* *hydrogen) + produces(X, energy)* $\rightarrow$ *fuel_cell(X).*

Terms which appear in both definitions form part of the abstract definition, such as *produces(X, energy).* Predicates which appear in both definitions but with different arguments require the introduction of a variable, such as *waste(X, W).* Introducing a variable captures a weaker similarity between the two concepts and is akin to weakening the pre-conditions, compared to either target concept.

The combination of these common factors yields the definition of the *source* abstraction: *requires(X, F)* $\wedge$ *waste(X, W)* $\wedge$ *produces(X, energy)* $\rightarrow$ *source(X).*

The derived abstraction sits in support of using the target predicate in place of the source. Swapping the predicate may result in the generation of rules when reaching an impasse. In this example the rule for the battery can be expanded to apply to a fuel cell. This is shown in Figure 1.

Two approaches to applying the analogy are: Firstly, the derived concept can replace battery in the explanation and a more general rule can be derived: *source(X) + max_output(X,O) > demand(C,Y)* $\rightarrow$ *can_supply(X,C).* Secondly, the *source* predicate could be replaced with *fuel_cell* and the analogy sits in support of this decision. The second approach is adopted in this paper.

An alternative to implicitly linking two terms and replacing one with the other is possible. Both terms could be replaced with the abstraction, which would be a more explicit analogical link. However, this could lead to over-generalisation. By storing a new rule which includes the abstraction, more than the two concepts used to derive it could now be applicable. It could be that analogies only hold in certain cases; the impact of this can be lessened by using each abstraction in only the situation which caused its derivation. Further work could be conducted in studying the over-generalisation issue with regards to this technique. Further nuances of this algorithm merit discussion.

3.3. Discussion

One disadvantage of applying an analogy is the utility problem. The *Utility problem* is defined as *"The difficulty in ensuring that an acquired concept enhances the performance of the application system"* (DeJong, 2004) and is discussed more in (Steven, 1990). Approaches to mitigating this should be employed, such as including *a priori* knowledge relating to utility in the domain theory. Another approach is the empirical testing and removal of analogies that increase the branching factor significantly without deriving useful rules.

Applying an analogy will have a cost, not just in matching but by increasing the branching factor of the problem. This is due to augmenting the domain theory; more possibilities result in longer searches. It may be that no useful information is gained by permuting a rule to apply to a new situation via analogy. There is no deductive evidence that the similarities between two concepts mean that a given rule applies to both. As there is a cost and potentially no gain, since the leaps are inductive, this technique should not be overused. There may be reason to think that similarity with nodes higher in an explanation structure are better for judging applicability. This is described in (Carbonell, 1983). There may also be reasons to prefer more specific rules (Huhns & Acosta, 1988). However, when no deductive option is available this technique may give an alternative to backtracking or failure. Choosing to derive an analogy could be likened to strategically weakening preconditions, given that an analogy is a subset of atoms of a concept.

It may be possible, at any point, to derive many potential analogies between concepts. However, since not all of these will be useful a conservative approach is to only adopt an analogy as a last resort to failure. If an analogy is adopted during an explanation to prevent failure, then the validity of the analogy could be assessed by whether the new rule can be successfully applied. The criteria for judging the utility of the rules may well be domain independent.

There may be more than one possible analogy which can be derived. Biases could be used to narrow down the list of potential definitions, ideally to one. One bias could be: A definition which makes more claims than another is stronger. Other biases could be derived by considering lexical similarity or using meta-data labelling of concepts. The bias employed in this paper is to simply take the analogy that makes the claim with the largest number of similarities. Analogues are computed for the potential matches and the strongest is chosen.

The source definition is given by backward-chaining from the goal which causes an impasse. The target is inferred to exist and must be searched for. This search is potentially expensive though only a subset of the domain theory could fit the pattern.

Another use for the derived analogy is when forming new rules, the analogy can be considered first and a general rule formed initially. The rule could also be annotated with any concepts considered during application for which it does not hold, like a caveat. This would follow from the assumption that these analogies are a simple representation of a potentially complex relationship and can be both correct and incorrect based on factors not modelled in the system.

Any analogy proposed by this method may be flawed since it is an attempt to infer some commonality and relationship that is not explicitly reasoned. The more this is applied, the greater the chance of concluding a fallacy. Therefore analogies should initially be proposed and should never be assumed to be correct. The argument is similar to that in (Dejong, 2006); these analogies capture information which can be both correct and incorrect depending on factors that the system is unaware of. To discard them too readily risks missing important relational information but they themselves may only capture an aspect of the real relationship. If an analogy never leads to useful rules that hold when encountered in reality it should be discarded. Since it may be impossible to know this, a probability threshold, or similar technique, could indicate which rules to discard. Alternatively, the comparison of the impact of a given analogy on the goals of the system could be used.

There is an issue with assuming an abstract similarity exists. Not all analogical inferences are equally likely (Davies, 1987) and where the similarity is based on features, not all will be relevant. This is partially ameliorated by EBL disregarding features not relevant to the example, using an inductive bias. When trying to derive an abstract similarity between two concepts, even using an example as an inductive bias, it is likely that not all features in the example are relevant to the analogue being derived. Using an example as an inductive bias, along with empirical validation, could help to disregard spurious analogies.

Commonality in language is important to this method as it requires either shared terms or analogous terms. However, commonality in language may not be an unreasonable assumption. This is because the knowledge was originally represented using an altered subset of a shared language between experts. Therefore it may be that lexical similarity can be used heuristically to establish or evaluate an analogue.

Note that multiple applications of analogy could build a common set of terms from other concepts, whilst discarding language that isn't commonly employed. The common terms could be the result of previous analogical reasoning. The effect of this technique would be most obvious where the domain theory uses disparate terms. In this type of domain, these analogy concepts can be used to transform nodes into a more common language.

There is a danger of applying this technique too readily since over-application would be costly and may bring about no real benefit. It is important to select when to use analogy since each application represents an inductive leap and weakens the otherwise deductive proofs EBL produces. It is important to bear these issues in mind during knowledge engineering.

The algorithm enables the expansion of a rule for the battery to consider a fuel cell, for example. However, whilst this looks like a sensible application of a rule in an abstract way it is likely that over-generalisation can occur. Alternatively, rules could be generated which would ideally be prohibited with a more complete domain theory. It is therefore important to only propose an analogy when the result is a useful rule. The utility of an analogy is not apparent at the time of derivation therefore the individual rules may need to be evaluated post generation. The algorithm is applied in simulation in order to explore these issues further.

4. Experiment Design

An experiment is proposed in order to show that analogy can allow EBL to generate fuzzy rules for a piece of hardware that is not catered for by the domain theory. The generation of fuzzy rules which reflect the intent of the system would satisfy this objective i.e. rules which reduce energy consumption by controlling previously unseen hardware.

The domain theory determines what rules can be generated. In this experiment it is designed for throttle control. However, the goal has had variables bound such that an impasse will be reached when attempting to generate rules for throttle control. This is because the variable binding *height_ to_ go* identifies a piece of hardware relating to pitch, rather than the throttle control. Analogical reasoning is required in order to generate control laws for pitch. Rules derived by this algorithm extend the deductive closure of the domain theory.

The generation of any rules which reduce energy consumption fulfil the intent of the original domain theory. The rules generated in this chapter control the pitch of the aircraft, which is not possible using the given domain theory without analogy. Therefore rules which reduce energy consumption generated in this experiment fulfil the second objective of this paper.

The system being tested is the same as in (Timperley, 2015). EBL was used to generate fuzzy rules which were executed in order to control the throttle of an aviation platform. The generation of useful rules, specialised from a body of general expert knowledge, was demonstrated by said controller requiring less energy to complete a set flight. This paper uses the same experimental setup as the previous work.

The approach presented in this paper has been implemented and integrated with the X-Plane simulator. X-Plane has previously been used in high-fidelity aerospace research, such as (Cameron et al., 2011), as part of a broader simulation suite.

Pitch changes are executed by the FLCHG system in X-Plane. This means that the pitch controls themselves are not used. It is the throttle which controls climb and descent. The throttle setting is calculated based on the height to the next checkpoint being positive or negative. Controlling the pitch via the joystick results in a single large alteration which is then compensated for, when possible, by the autopilot. Because of this, it is the height to the next checkpoint which is controlled in this paper. Alteration of the required climb indirectly affects the throttle and therefore the pitch.

An experiment was conducted in order to evaluate the approach presented in this paper. A series of replicable flights were devised to test the controller. Each flight begins with a take-off, followed by a series of checkpoints to be hit. The run ends when the final checkpoint is reached. The checkpoints are placed such that climbing, descending and turning all feature. The autopilot flies a set route, input into a Flight Management System (FMS).

There are two sources of variation between the flights within a data set. The first is the throttle control value, which is altered by the proposed algorithm. This is allowed to vary over its entire value range [0-1]. The second source of variation is the simulator. The time of day is not bound which may affect temperature and therefore air pressure. This can alter the amount of energy perform certain activities, such as take off. The weather was set to be ‘clear’ which limits the variability of the weather significantly which in turn limits the effect on the simulated flights. Weather includes the winds encountered and other sources of pressure variation. Limiting these effects limits the variation in forces encountered during. This maintains a level of consistency in the energy requirements of each flight. The remaining variables controlled by the simulator are bounded by keeping the route fixed. The domain theory persistent between flights. A data set uses a single domain theory which is updated in each flight and passed to the next.

The domain theory only contains rules to determine state, reason about relative orders of magnitude, and reduce the throttle of the aircraft. There are no rules to control pitch. The domain theory is constructed in a way which could support analogical reasoning. The characteristics required of such a domain theory are: additional contextual information which can act as definitions for source and target concepts. There should also be sufficient generalisation that source predicates can be replaced.

For example, consider a system input being identified by the statement *velocity(input_ value)*. Searches for *X(input_ value)* would identify predicates with a particular input value. However, if the same system input is identified by *reporting(velocity, input_ value)* then target concepts can be identified by the name of the input, rather than the current value.

The inputs and the output are abstracted such that predicate replacement can occur. Additionally, the output is no longer hard coded into a parameter. This could aid the flexibility of the domain theory in a deductive sense. By abstracting the inputs and output, the relationship is made to apply to different sources.

Changes were made to the domain theory to support the formation of specific analogies. Rules which have shared terms have been added to the domain theory to support the generation of analogies. These rules embody the additional contextual information about concepts. This additional information, unnecessary to the intended model, may be easier than eliciting a minimal domain theory. It is worth noting that extra information may not always need to be added in order to form analogies, but can be used to restrict the analogies which are possible. This can be a helpful tool in reducing unwanted rules from being derived.

The domain theory is given in Table 2. The table also includes the goal used to derive rules.

Goal
reduce(height_to_go,Y)
Domain Theory:
Rules
numbers
bigger(X,Y) → smaller(Y,X)
!same(X,Y) → different(X,Y)
!bigger(Y,X) → biggest(X)
!smaller(Y,X) → smallest(X)
!same(X,Y) ∧ bigger(X,Z) ∧ bigger(Z,Y) → bigger(X,Y)
reporting(I,X) ∧ bigger(X,Y) → more_than(I,Y)
reporting(I,X) ∧ smaller(X,Y) → less_than(I,Y)
reporting(I,X) ∧ same(X,none) → none(I)
more_than(I,none) → some(I)
reporting(I,X) ∧ same(X,Y) → is(I,Y)
reporting(I,X) ∧ started(I,Y) ∧ same(X,Y) → unchanged(I)
reporting(I,X) ∧ started(I,Y) ∧ !same(X,Y) → changed(I)
reporting(I,X) ∧ smaller(Y,X) → one_smaller(I,Y)
reporting(I,X) ∧ bigger(Y,X) → one_bigger(I,Y)
reduction strategy
reporting(I,X) ∧ intend(I,N) ∧ bigger(X,N) → reduce(I,N)

throttle_command(I) ∧ height_to_go(H,I) ∧ velocity_sensor(C,I) ∧ reporting(H,P) ∧ reporting(C,E) ∧ same(P,E) ∧ one_smaller(I,N) → intend(I,N)
throttle_command(I) ∧ height_to_go(H,I) ∧ velocity_sensor(C,I) ∧ reporting(H,P) ∧ reporting(C,E) ∧ smaller(P,E) ∧ one_smaller(I,N) → intend(I,N)
state
some(parking_brake) ∧ less_than(throttle,medium) ∧ none(altitude) → state(idle)
none(parking_brake) ∧ less_than(throttle,medium) ∧ none(altitude) → state(taxi)
none(parking_brake) ∧ more_than(throttle,small) ∧ none(altitude) → state(takeoff)
current(takeoff) ∧ none(parking_brake) ∧ some(altitude) → state(flight)
analogy
controls(engine_speed) ∧ affects(air_speed) ∧ affects(acceleration) ∧ provides(thrust) ∧ reflects(power) → throttle_command(throttle)
controls(elevators) ∧ affects(altitude) ∧ affects(air_speed) ∧ pitches(aircraft) → pitch_command(height_to_go)
reflects(motion) ∧ affected_by(throttle) → velocity_sensor(velocity, throttle)
reflects(climb) ∧ affected_by(pitch) ∧ affected_by(throttle) → climb_sensor(climb_rate, height_to_go)
change_in(y) ∧ affected_by(throttle) ∧ affected_by(pitch) → height_to_go(height_to_go, throttle)
change_in(x) ∧ affected_by(throttle) ∧ affected_by(pitch) → ground_speed(groundspeed, height_to_go)
Statements
numbers
bigger(full,huge)
bigger(huge,large)
bigger(large,medium)
bigger(medium,small)
bigger(small,tiny)
bigger(tiny,none)
same(X,X)
analogy
throttle_command(throttle)
pitch_command(height_to_go)
height_to_go(height_to_go, throttle)
velocity_sensor(velocity, throttle)

Table 2: Analogical Domain Theory

The parameter values in the domain theory are used to identify simulator inputs. During explanation these cause variables to be bound to the current reading for the identified input. Examples of these parameter values include:

throttle, *velocity_ sensor*, *height_ to_ go*, *parking_ break*, *altitude*. Some bound values in the analogy section refer to abstract notions that do not refer to system inputs; pitch is one of these. These represent disconnected background information derived from the expert which add context but aren't directly related to rule generation without analogy. They can be altered to altered to influence the abstractions which can be generated.

The use of a single goal for this experiment also serves to limit the possible analogies. This leaves the domain theory with an inherently hierarchical structure.

It is worth noting that, in the current implementation, analogies are not added to the domain theory as new rules. The (sub)goal causing an impasse is swapped for the target with the larger correspondence i.e. the one which formed the most complex derived class. Additionally, the analogous nodes are not included in the fuzzy rule derivation process. These nodes are omitted because the nature of the link between the source and target terms is assumed to be irrelevant to the fuzzy rule. Instead the analogy is viewed more as evidence, or an operational node in support of the swapped node. This allows the explanation to continue without adding goals which would not have been present in a non-analogical explanation structure, minus the term that has been replaced.

When attempting to use EBL to derive a new fuzzy rule an impasse will be reached, because there is no deductive explanation with the given domain theory for pitch control. These impasses can be avoided by forming analogies. The analogies in this case replace the inputs being compared and the output, but maintain the same overall structure as the throttle derivation from the previous work. The output is swapped from the throttle to the height to the next checkpoint. The inputs were also altered. The velocity sensor is swapped for the height to go to the next checkpoint. The height to go is swapped for ground speed.

Swapping the inputs follows a simple rationale. When the ground speed is smaller than the height to go to the next checkpoint, reduce the climb rate. The climb rate could be calculated as the gradient: $\frac{\delta y}{\delta x}$. A larger gradient, and steeper climb, would come about when the change in x (δy) is larger than the change in y (δx). Therefore reduction occurs when ground speed is smaller than the height to go to the next checkpoint.

Analogy is required for the generation of all fuzzy rules during this experiment. No rules can be generated by the system initially, analogy is used to swap the output and inputs during rule generation. The domain theory remains largely the same as in the previous work. Some small additions are made to facilitate the formation of certain analogies. By swapping the output and inputs a general strategy is transferred to control an item of hardware which lies outside the original domain theory. This can be seen as further relaxing the restrictions that were present in the rule generation method proposed in the previous work. This has been facilitated by swapping the inputs and outputs referenced in a rule derivation.

Inputs are fuzzified in the manner proposed in the previous work. Each measurement is fuzzified and interpolated and is therefore relative to itself.

This is because the range over which the linguistic values are interpolated is determined by past values. The rules generated are dependent on the number of fuzzy sets and refresh rate of inputs. Both of these factors influence the states which are considered by EBL, as well as the granularity of each rule.

The analogies formed, by the proposed algorithm, in order to swap the inputs and output are given below. The derived concept is labelled as the target concept.

- affected_by(throttle_ratio_all) $\wedge$ reflects(R)$\rightarrow$ climb_sensor(climb_rate, height_to_go)
- affected_by(pitch) $\wedge$ change_in(C)$\rightarrow$ ground_speed(groundspeed, height_to_go)
- affects(A) $\wedge$ controls(C)$\rightarrow$ pitch_command(height_to_go)

Note that the output is taken as an input. This output is also written to through the switching mechanism. Having one input being written to by two systems means that timing can affect the rules learned, since the input will appear to have one value or another, depending on when it is sampled. This is one motivation for using fairly large data sets and averaging. It is worth noting that in this experiment the autopilot is always operated at 10Hz and the controller is initially operated at the same rate.

In this experiment the outputs of forming an analogy are any fuzzy rules generated by employing said analogy. As no rules can be formed deductively, all rules will be formed in this experiment by analogy. Any rules generated can, depending on their impact on the flight, sit in support of the technique. The aim of the experiment is to show the possibility that the technique can be used to generate useful rules after knowledge engineering has taken place. The results of the experiment are presented in the next section.

5. Results and Discussion

A set of 50 flights was conducted with a set series of waypoints. The goal used in this experiment leads to the generation of rules which control pitch. However, no rules can be generated for pitch control, using the domain theory designed for throttle control, without resolving impasses. In order to satisfy the first objective of this paper rules must be generated after reaching impasses. The second objective can be satisfied by demonstrating the production of useful rules.

There are two sources of variation between the flights within a data set. The first is the throttle control value, which is altered indirectly by analogically derived rules via the *height_to_go* to the next checkpoint. The *height_to_go* is used by the autopilot to determine throttle but itself relates to pitch.

The second source of variation is the simulator. The time of day is not bound which may affect temperature and therefore air pressure. This can alter the amount of energy perform certain activities, such as take off. The weather was set to be ‘clear’ which limits the variability of the weather significantly which

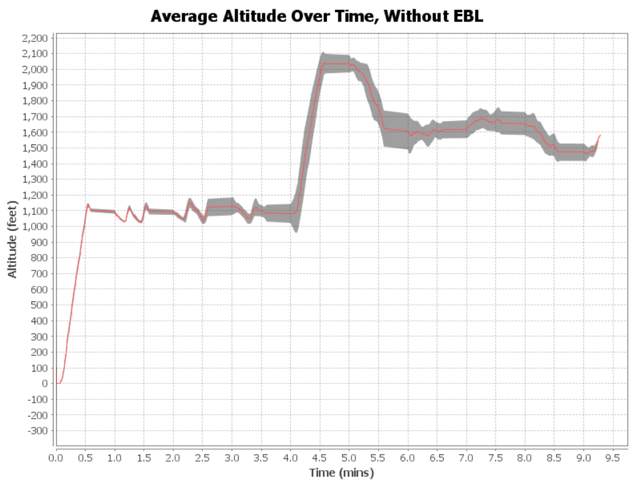

Figure 2: An average, over 50 flights, of the altitude of the aircraft. This is without Analogy. The shaded areas indicate the standard deviation around the mean. The inputs were sampled at 10Hz.

in turn limits the effect on the simulated flights. Weather includes the winds encountered and other sources of pressure variation. Limiting these effects limits the variation in forces encountered during. This maintains a level of consistency in the energy requirements of each flight. It may be possible to further limit the weather and time variability using X-Plane. The remaining variables controlled by the simulator are bounded by keeping the route fixed. The domain theory persistent between flights. A data set uses a single domain theory which is updated in each flight and passed to the next.

There are two reasons to consider the average altitude of the flights within a dataset. Firstly, the consistency of the dataset can be shown by considering the variation in the dataset without analogy, where no rules can be generated. Secondly, by comparing the average altitudes of each dataset, with and without analogy, significant impacts that the generated rules have on flight can be seen. This establishes that the generated rules impact the system. After which, it remains to demonstrate that the rules positively impact energy conservation.

The mean altitude for all 50 flights, along with the standard deviation, are plotted in Figure 2. The small standard deviation implies that the route followed was consistent, with small differences, across all 50 flights.

The standard deviation shows the areas most affected by the derived rules. Large standard deviations can be caused by differences in aircraft behaviour at the same point on the time axis. As can be seen, the effect is concentrated at the start of a climb and in the later section of flight (from about 5.5 minutes to 7 minutes).

The controller's effect being concentrated at the start of a steep climb may be a result of the inputs being measured relative to themselves, rather than a scale that applies to both. The rate of increase in either input may be partially

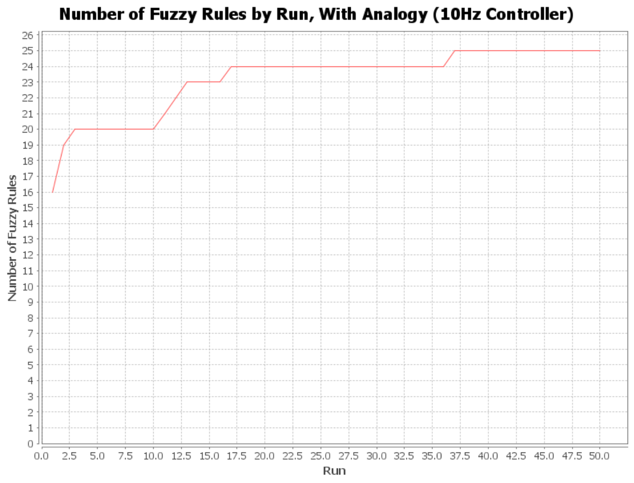

Figure 3: The number of rules generated by the end of each flight is given. Rules generated in one run are not guaranteed to be executed on the same run. This graph is for the data set collected using analogy and the inputs were sampled at 10Hz.

hidden. When both inputs are increasing then a small change, relative to its own scale, might be significant for an input. However, it is possible such an increase is not significant to the relationship being appealed to in the domain theory. Additionally, the baseline throttle control is rather binary. A binary throttle controller leads to a high ground speed when climbing, even when the climb is steep. Investigation of an interpolated fuzzy relationship for the comparison of two inputs could prove an interesting area for future work.

The initial section of the steep climb is reduced drastically, then returns to a steep climb. This is less granular control than was exhibited when applying the domain theory to throttle control (Timperley, 2015). The loss of granular control is related to the throttle output only being effected in a binary manner, based on the finer control of another output (height to go). This means that a reduction in pitch may lack the finer control required to approach optimality.

The effect of the rules, from these two examples, appears to be a general reduction in extreme levels of throttle output. However, more fluctuations are present. Fluctuations could cause an increase in the distance travelled without saving sufficient energy to have a net positive effect. Also, some of the larger plateaus have fluctuations which are small. These fluctuations come from the effect of the fuzzy rules competing with the auto-throttle. This shows that the effect of the fuzzy controller may be truncated by competition with the auto-throttle. The degree of competition with the auto-throttle is affected by the switch timings.

The number of rules generated, by run, is shown in Figure 3. A full listing of the rules generated is given in Table 3. Each of these rules required analogical reasoning during derivation. They satisfy the first objective of the paper.

Run	**Input**			**Output**
	ground speed	**climb rate**	**Height to go**	**Height to go**
1	large	large	small	tiny
1	medium	medium	medium	small
1	huge	huge	medium	small
1	huge	huge	huge	large
1	small	small	medium	small
1	medium	medium	tiny	none
1	medium	medium	small	tiny
1	small	small	tiny	none
1	huge	huge	large	medium
1	huge	huge	tiny	none
1	small	small	huge	large
1	huge	huge	small	tiny
1	large	large	medium	small
1	large	large	large	medium
1	none	small	medium	small
1	small	small	small	tiny
2	medium	medium	huge	large
2	large	large	tiny	none
2	medium	medium	large	medium
3	large	large	huge	large
11	medium	medium	medium	none
12	small	small	large	medium
13	medium	medium	small	none
17	medium	medium	huge	small
37	large	large	medium	tiny

Table 3: Table of the fuzzy rules generated, by run, with analogy. The controller was operated at 10Hz.

The sharp increase of rules in the final run may indicate that more rules may have been generated given a larger number of runs. The fuzzy rules, generated by analogical EBL, affected the remaining battery charge. Battery charge measures the mainstay of the energy available to the platform. A more efficient flight results in a higher remaining charge at the end of a run. If the controller generates useful rules then a more efficient flight will be indicated by remaining charge. The generation of useful rules via analogy satisfies the second objective of the paper. The remaining battery charge is plotted by run in Figure 4.

Figure 4 shows that some rules had a positive effect on efficiency but that the gains are situational. The rules are generated by analogy as a first attempt to extend an incomplete domain theory. Each rule represents a shallow understanding of a deeper concept. The rules should ideally use empirical data about

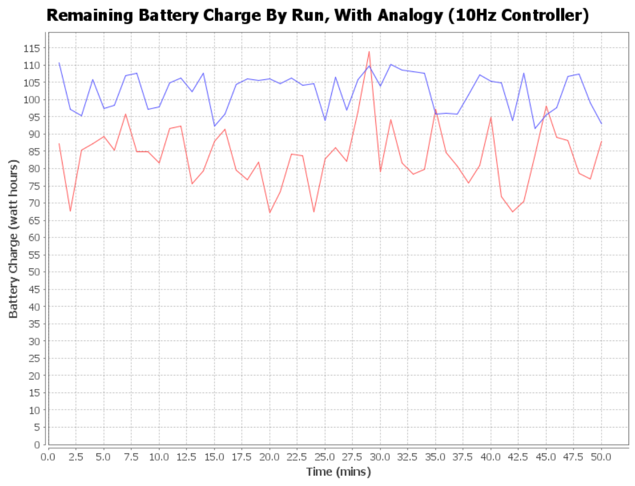

Figure 4: The battery charge remaining at the end of each run for flights both with (red) and without Analogy (blue). Inputs sampled at 10Hz.

individual rule impact, as well as specific combinations of rules in different orders. This data could be used to further specialise the rule to better reflect a deeper understanding of the concept it represents.

Rules impact is situational and examples of a negative impact can be seen in Figure 4. This is likely due to the large reduction in altitude between about 5.5 and 7 minutes. This results in a longer path taken to reach the final checkpoint. A longer route in this case costs more energy than is saved by reducing altitude.

The fuzzy rules are all part of a reduction strategy, which sits in opposition to the auto-throttle ascending to a new checkpoint. The switch settings alter the balance between these two factors. The effect of the switch timings is further examined in a series of flights using different sampling rates for the controller.

Sets of 50 flights were again conducted. In each case the switch setting for the fuzzy controller was altered. The remaining charge, both with (red) and without Analogy (blue) is given, as well as the number of rules generated per flight. The aim is to elicit any general trends between the switch settings, the efficacy and the number of rules generated. The following switching rates are presented: 6.6Hz (every 0.15 seconds), 5Hz (every 0.20 seconds), and 4Hz (every 0.25 seconds). The figures for each result set are given below:

- The number of rules generated for a 6.6Hz controller is given in Figure 5. The remaining battery charge is shown in Figure 6.
- The number of rules generated for a 5Hz controller is given in Figure 7. The remaining battery charge is shown in Figure 8.
- The number of rules generated for a 4Hz controller is given in Figure 9. The remaining battery charge is shown in Figure 10.

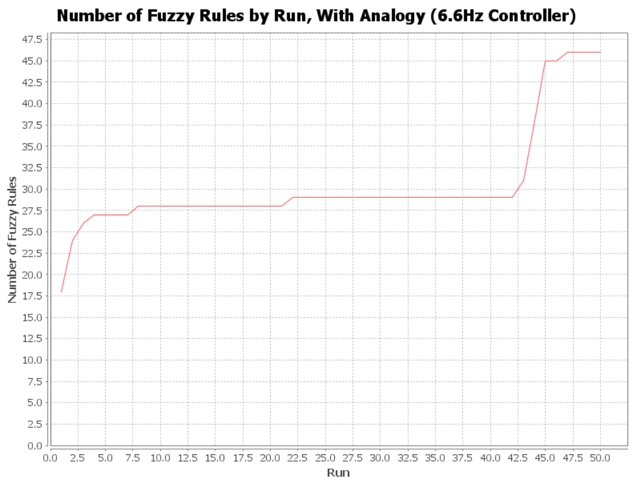

Figure 5: The number of rules generated by the end of each flight. Analogical controller enabled, inputs sampled at 6.6Hz.

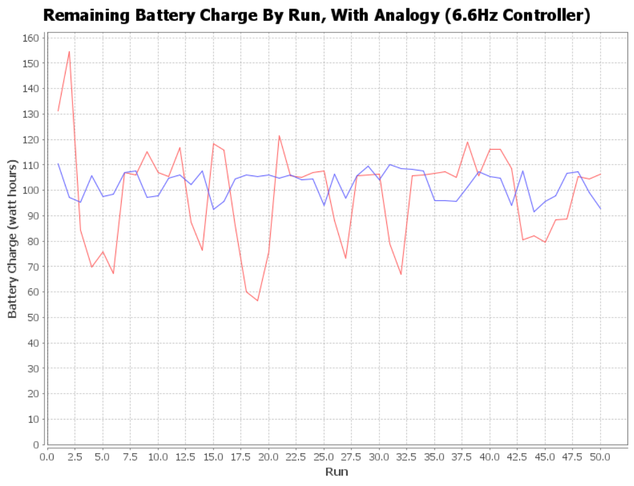

Figure 6: The battery charge remaining at the end of each run. Inputs sampled at 6.6Hz.

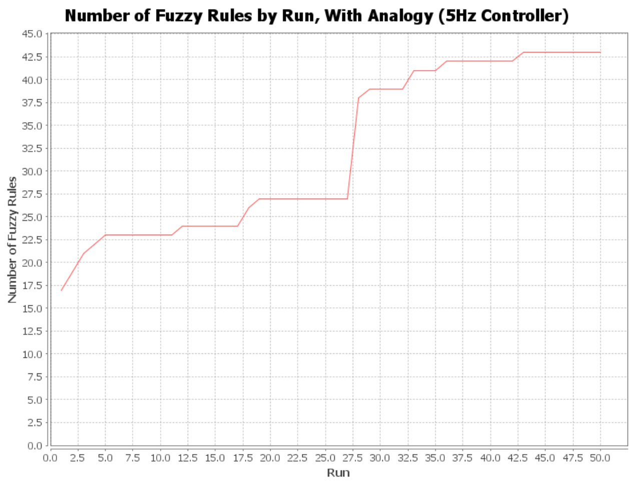

Figure 7: The number of rules generated by the end of each flight. Analogical controller enabled, inputs sampled at 5Hz.

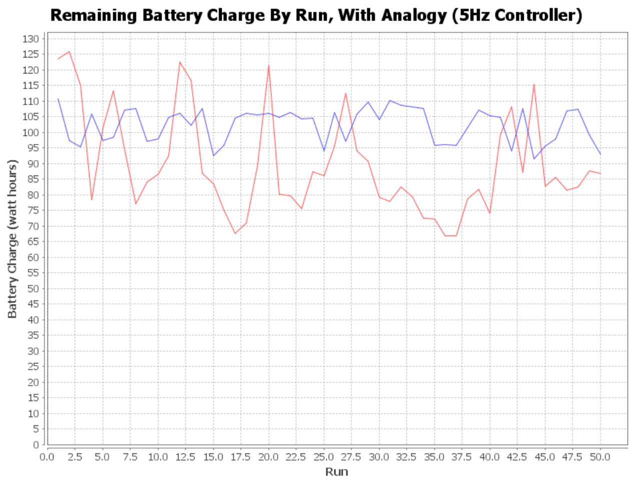

Figure 8: The battery charge remaining at the end of each run. Inputs sampled at 5Hz.

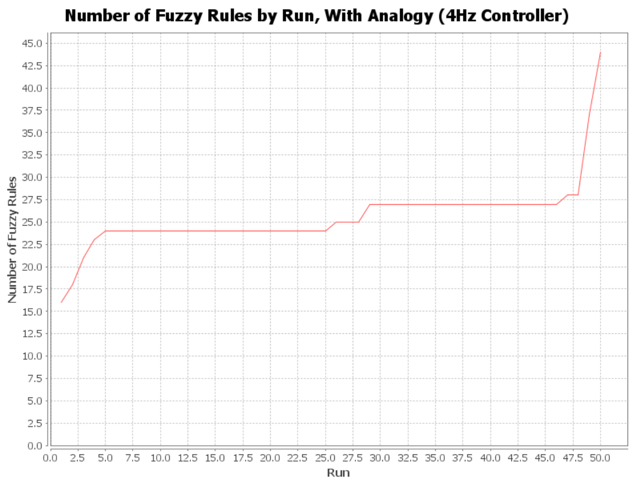

Figure 9: The number of rules generated by the end of each flight. Analogical controller enabled, inputs sampled at 4Hz.

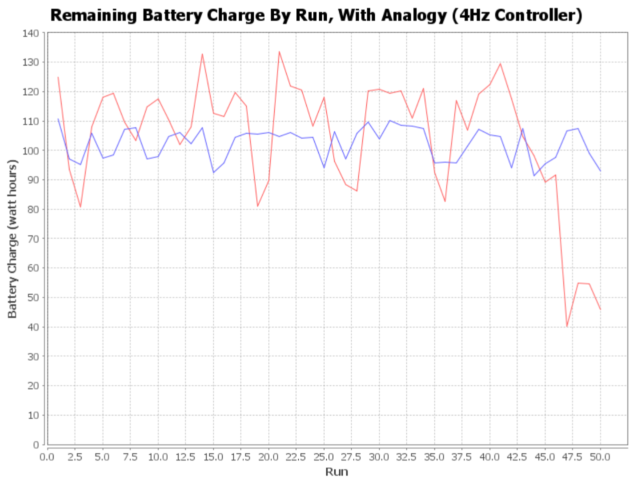

Figure 10: The battery charge remaining at the end of each run. Inputs sampled at 4Hz.

The 'height to go' input, which is also the output, is written to by both the auto throttle and fuzzy controller. Rule generation is guided by the input value. This balance may simply have exposed the EBL algorithm to more situations than other sampling rates.

More rules were generated when the controller operated at the highest frequencies. Rule generation was most prolific at 6.6Hz. Operating the controller at a higher rate gives it a higher resolution. Rules can be executed which reduce pitch, which prevents the aircraft reaching a state which necessitates the generation of more rules. With a large pitch several rules need to be generated in order to result in a reduction in pitch. This is because a change is only affected by the autopilot once the output becomes negative, so several rules are required to bring a large pitch to a negative value. By keeping the pitch at smaller values, fewer rules are required. This may point to higher rates of controller execution being beneficial.

Some rules may come from seeing an output from a previous controller execution, rather than the value from the autopilot. The input may be insufficiently reduced to cause a change in pitch and the algorithm may produce further rules as a consequence. Not all rules generated are guaranteed to affect the throttle on their own, since the auto throttle is specified as being concerned with the sign of the value. Therefore several rules may need to be generated in turn, and executed in concert, to result in an effect. This could complicate a study of the impact of individual rules in this case.

The remaining charge was generally improved by the 4Hz controller, but a significant loss in efficiency was evident in later runs. This may imply that rules generated earlier in this case were of a greater utility. This implies that even when useful rules are generated from an analogy, invalid rules may also be generated. An algorithm for adding caveats to an analogy in order to prevent detrimental rules from being learned could be a fruitful aspect of further work. This would likely require clearer data on the individual impact of each rule, both alone and in conjunction with others. Both cases need to be considered, as multiple rules may combine to produce a positive effect where only one would not.

6. Conclusion

An approach for incorporating analogy into EBL has been presented. The technique could be used to further generalise rules in a system by linking previously separate concepts. These links are formed by deriving an abstraction. Doing this may imply that whatever reasoning held in deriving the original rule analogously holds for the new hardware. Analogy is an example of transfer learning.

Transfer learning can be applied as an alternative to an impasse. This technique can operate where an impasse is reached due to a mismatch between outer predicates with the same form. The reason for addressing this type of impasse is that EBL, by generalising, already avoids some potential impasses due to differing parameters.

By applying this technique EBL can make inductive leaps when deductive explanation fails. However, this entails all of the drawbacks of making inductive leaps. Given that the potential costs are weighed against the possibility of no gains, the technique should be applied conservatively.

However, flights which performed significantly below the baseline could imply that the technique requires further work. Each rule should ideally be evaluated for its individual impact, but where the derivations are only possible by analogy it is likely that there may be no information for evaluation. Deviation from known mission parameters or principles could be possible. Further work into eliciting a general evaluation strategy for analogies would be beneficial. It would be particularly beneficial to glean more from the underlying correlation than just a connecting analogy. Each analogy holds only in certain circumstances. For this reason, formation of analogy may be considered the start of an ongoing learning task.

Some rules were generated which satisfy the aim of this paper, by generating rules for a piece of hardware which was not within the original deductive closure of the system. However, the nature of the results illustrate that rules other than ones with a positive impact on energy management were produced. This means that the intent of the domain theory was not realised in every rule generated. The discrimination of useful rules should be the focal point of further work.

7. Acknowledgements

The authors would like to thank EPSRC for funding this work.

References

Burstein, M. H. (1986). A Model of Learning by Incremental Analogical Reasoning and Debugging. *Machine Learning: An Artificial Intelligence Approach (Vol. 2)*, .

Cambria, E., Fu, J., Bisio, F., & Poria, S. (2015). AffectiveSpace 2: Enabling affective intuition for concept-level sentiment analysis. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (pp. 508–514). Austin, Texas: AAAI Press.

Cameron, N., Webster, M., Jump, M., & Fisher, M. (2011). Certification of a Civil UAS: A Virtual Engineering Approach. In *2011 AIAA Modelling Simulation and Technologies Conference and Exhibit* (pp. 1–15). Portland, Oregon: AAAI Press.

Carbonell, J. G. (1983). Derivational Analogy and Its Role in Problem Solving. In *AAAI-83*.

Committee on Materials, Structures, and Aeronautics for Advanced Uninhabited Air Vehicles, Commission on Engineering and Technical Systems, N. R. C. (2000). *Uninhabited Air Vehicles : Enabling Science for Military Systems*. Washington, DC, USA: National Academies Press.

Davies, T. R. (1987). A logical approach to reasoning by analogy. In *In IJCAI-87* (pp. 264–270). Morgan Kaufmann.

DeJong, G. (2004). Explanation-Based Learning. In A. Tucker (Ed.), *Computer Science Handbook*. (2nd ed.).

Dejong, G. (2006). Toward Robust Real-World Inference: A New Perspective on Explanation-Based Learning.

Ellman, T. (1989). Explanation-based learning: a survey of programs and perspectives. *ACM Comput. Surv.*, *21*, 163–221. doi:`http://doi.acm.org/10.1145/66443.66445`.

Faculty, T. A., & Kahn, D. (2001). The Design and Development of a Modular Avionics System.

Falkenhainer, B. (1987). An examination of the third stage in the analogy process: verification-based analogical learning. In *Proceedings of the 10th international joint conference on Artificial intelligence - Volume 1* IJCAI'87 (pp. 260–263). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, *7*, 155–170.

Hirowatari, E., & Arikawa, S. (1994). Incorporating Explanation-Based Generalization with Analogical Reasoning. *Bulletin of Informatics and Cybernetics*, *26*, 13–33.

Huhns, M. N., & Acosta, R. D. (1988). Argo: a system for design by analogy. doi:`10.1109/64.21891`.

Ishikawa, T., & Terano, T. (1996). Analogy by abstraction: Case retrieval and adaptation for inventive design expert systems. *Expert Systems with Applications*, *10*, 351–356.

Klenk, M., & Forbus, K. (2009). Domain transfer via cross-domain analogy. *Cognitive Systems Research*, *10*, 240–250.

Könik, T., O'Rorke, P., Shapiro, D., Choi, D., Nejati, N., & Langley, P. (2009). Skill transfer through goal-driven representation mapping. *Cognitive Systems Research*, *10*, 270–285.

Lopez, J., Royo, P., Barrado, C., & Pastor, E. (2008). Modular avionics for seamless reconfigurable UAS missions. doi:`10.1109/DASC.2008.4702748`.

de Mántaras, R. L., & Plaza, E. (1997). Case-Based Reasoning: an overview. *AI Commun.*, *10*, 21–29.

Mooney, R. J., Bennett, S. W., & Urbana, A. (1986). A DOMAIN INDEPENDENT EXPLANATION-BASED GENERALIZER. In *AAAI-86* (pp. 551–555). Philadelphia, Pennsylvania: AAAI Press.

Steven, M. (1990). Quantitative results concerning the utility of explanation-based learning. *Artificial Intelligence*, *42*, 363–391.

Timperley, M. (2015). *The Integration of Explanation-Based Learning and Fuzzy Control in the Context of Software Assurance as Applied to Modular Avionics*. Ph.D. thesis University of Central Lancashire.

VanLehn, K., & Jones, R. M. (1998). Learning Physics Via Explanation-Based Learning of Correctness and Analogical Search Control,.

Wilson, A., & Preyssler, T. (2009). Incremental certification and Integrated Modular Avionics. *Aerospace and Electronic Systems Magazine, IEEE*, *24*, 10–15. doi:10.1109/MAES.2009.5344176.