Cross-domain Correspondences for Explainable Recommendations

Aaron Stockdill
Daniel Raggi
Mateja Jamnik
aaron.stockdill@cl.cam.ac.uk
daniel.raggi@cl.cam.ac.uk
mateja.jamnik@cl.cam.ac.uk
University of Cambridge, UK

Grecia Garcia Garcia
Holly E. A. Sutherland
Peter C.-H. Cheng
g.garcia-garcia@sussex.ac.uk
h.sutherland@sussex.ac.uk
p.c.h.cheng@sussex.ac.uk
University of Sussex, UK

Advait Sarkar
advait@microsoft.com
Microsoft Research
Cambridge, UK

ABSTRACT

Humans use analogies to link seemingly unrelated domains. A mathematician might discover an analogy that allows them to use mathematical tools developed in one domain to prove a theorem in another. Someone could recommend a book to a friend, based on understanding their hobbies, and drawing an analogy between them. Recommender systems typically rely on learning statistical correlations to uncover these cross-domain correspondences, but it is difficult to generate human-readable explanations for the correspondences discovered. We formalise the notion of 'correspondence' between domains, illustrating this through the example of a simple mathematics problem. We explain how we might discover such correspondences, and how a correspondence-based recommender system could provide more explainable recommendations.

KEYWORDS

representation, correspondence, analogy, concept mapping

ACM Reference Format:
Aaron Stockdill, Daniel Raggi, Mateja Jamnik, Grecia Garcia Garcia, Holly E. A. Sutherland, Peter C.-H. Cheng, and Advait Sarkar. 2020. Cross-domain Correspondences for Explainable Recommendations. In *Proceedings of IUI workshop on Explainable Smart Systems and Algorithmic Transparency in Emerging Technologies (ExSS-ATEC'20).* Cagliari, Italy, 8 pages.

1 INTRODUCTION

When explaining a concept, people adapt their explanation for their audience [12]. This switch can be motivated by the concept itself, the audience's knowledge, or the motivation behind sharing the concept [11, 15]. Each factor must be balanced by the explainer before choosing a representation for the explainee.

Consider the problem of finding a closed form solution to the sum of integers between 1 and n. The 'formal' presentation might be

$$\sum_{i=1}^{n} i \tag{1}$$

which is compact and unambiguous. But it presupposes an understanding of higher mathematical notation, while providing no clues to the closed form solution. One might have to perform an induction, or identify an equivalent series and work algebraically with them; both are advanced techniques to reach the solution $n(n+1)/2$. This algebraic representational system (hereafter a *representation*) is *formally* appropriate, but may not be *cognitively* appropriate.

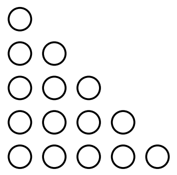
Figure 1: The numbers 1 through to 5 as rows of dots.

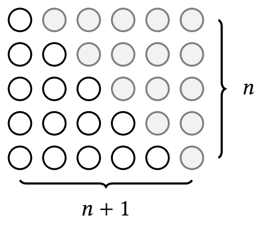

Figure 2: The general relationship for n dots from the case $n = 5$; by counting the whole grid and halving the result, we have $n(n+1)/2$ black-edged dots.

We can re-represent Equation (1) as in Figure 1, but instantiated to $n = 5$, where numbers are rows of dots. Modulo generalisation, these representations show the same key information, but their cognitive features are different: the dot diagram hints at the closed form solution whereas in the symbolic representation this is missing. Now, observing that we have a triangle, applying simple counting and reflection rules yields the solution in Figure 2.

Symbolic algebra and grids of dots are not equivalent representations: the former can encode many concepts that the latter cannot. Nor are the two forms of the problem equivalent: our remark 'modulo generalisation' exposes one difference. But the statements do capture the same *important properties* [13]: integers, associativity, commutativity, equality, etc.

A complication in preserving important properties across representations is that the properties take different forms. The symbol 2 becomes ◦◦, while + becomes stacking. Similar to analogy or structural similarity, we define *correspondences* [13] as links between *properties* which 'fill the same role' in their respective representations. Correspondences are thus both a heuristic for transformation between representations, and part of a justification for why a particular representation may be suitable for the given problem.

In this paper we explore precisely what correspondences are and how we can discover them, with the aim of helping users to discover and understand analogies. Our work is part of a pipeline to automatically recommend alternative representations to help users solve mathematics problems. We envisage a complete intelligent tutoring system capable of understanding the individual student, and adapting to their needs and preferences dynamically, for each problem they are tasked to solve. This paper represents one step towards this goal.

2 MOTIVATION

Analogical reasoning is a powerful tool [7]. Higher-order correspondences describing relationships allow deeper understanding of both the source and target statements by exploiting their structure [5]. From education to scientific discovery, insight occurs when disparate ideas are brought together [2, 6, 8, 16].

We previously introduced in [13] a rep2rep framework for automatically analysing and suggesting to users a suitable alternative representation for their current problem. The aim is for such intelligent systems to tailor this suggested representation to the needs of the person solving the problem. Our framework enables one to describe problems, representations, and the correspondences between them in a way that can be input to an algorithm which then produces the appropriate recommendation. The framework provides templates which analysts can fill with many kinds of values: types, tokens, patterns, and other properties. Types describe the grammatical roles in a representation, while tokens are what fills these roles. Patterns are 'conceptual groupings' of properties which form recognisable units for expert users. Analysts can associate attributes to properties: this could be types, descriptions, counts, or other information. Our framework is purposefully general: representations are extremely diverse, and we want to encode as many as possible.

Correspondences between representations are catalogued by analysts. This passes the burden of discovering analogies to a human ahead of time; when our algorithm suggests an appropriate representation, it selects relevant correspondences from this set. This simplifies the recommendation task considerably, but it leaves analysts with a formidable cataloguing challenge, so even partially automating this step is a high priority. We shall explore our approach to this problem in Section 4.

The final recommendation from the rep2rep algorithm[1] is a list of potential representations ordered high-to-low by their formal score. We define the formal score v_q^r for representation r to encode problem q as

$$v_q^r = \sum_{c \in C} s_c \cdot i_c^q \cdot \text{match}_q^r(c) \tag{2}$$

where C is the set of correspondences, s_c is the strength of correspondence c, i_c^q is the importance of c for problem q, and $\text{match}_q^r(c)$ is an indicator function equal to 1 when c is satisfied by the properties of q and r, and 0 otherwise. Intuitively, we are counting reasons why representation r is good, and weighting those reasons by both how important they are to the problem q, and how 'good' those reasons are in general (the correspondence strength s).

[1]The recommendation is presently based only on the formal properties; ongoing work incorporates the user profile into the decision.

Previous work around structure-mapping constructs correspondences between two analogous concepts by matching their internal associations [4]. This requires well-defined internal structure—in our digitally catalogued world, such a requirement is increasingly easily met. But structure-mapping assumes we already know that two concepts are related, we just need to work out *how* they are related. When we are searching for an analogy, we do not know this. Thus, the purpose of our research is to *discover* related concepts through analogy.

We have an implementation of the concepts discussed in this paper; the code is available at https://github.com/rep2rep/robin.

3 CORRESPONDENCES

In this section, we formalise the idea of correspondence first introduced in [13].

3.1 Transformations

A correspondence associates an encoding of content in one representation to an encoding of the same content in another representation, along with a measure of how well that content is preserved between them. When both source and target representations are specified, the correspondences describe how they are analogous; when only the source representation is specified, the correspondences specify *requirements*. In the example from Section 1, dots fulfil the requirements: they can encode integers and summation. A representation like the lambda calculus would also fulfil the requirements, but a Venn diagram representation would not. If instead our problem required encoding trigonometric functions, our dot representation and Venn diagram representations would fail to satisfy the requirements, and not be recommended.

Formally, we define a correspondence as a triple

$$\langle p_1, p_2, s \rangle \tag{3}$$

where p_1 and p_2 are formulae of properties and s is a real value between 0 and 1 [13]. The property formulae use the connectives AND, OR, and NOT with the expected interpretations. This allows for sophisticated relationships between representation properties. Value s denotes the *strength* of the correspondence: when $s = 0$, there is no correspondence, the content is unrelated; when $s = 1$, there is a perfect correspondence, the content is the same.

If we wanted to encode the correspondence 'a $\sum$ is like stacking *either* horizontally or vertically', we first identify the properties involved. Properties consist of a kind, a value, and attributes. In this case, we have a 'token' kind of property from an algebraic representation,

$$p_1 = (\text{token}, \textstyle\sum, a_1)$$

(with attributes abstracted to a_1 for this example), while we have two 'pattern' properties from the dots representation,

$$p_2 = (\text{pattern}, \text{stack-horizontal}, a_2)$$
$$\text{and } p_3 = (\text{pattern}, \text{stack-vertical}, a_3)$$

(where the attributes are similarly abstracted). The correspondence would thus be

$$\langle p_1, p_2 \text{ OR } p_3, 0.9 \rangle. \tag{4}$$

Note the key word *either* in our specification: these target properties are not independent, so we do not write two correspondences. Instead, we use a disjunction connective to make the sufficiency of one property or the other explicit. This is a strong correspondence, and hence has a high strength of $s = 0.9$.

Our definition of correspondences makes them *explainable*. Because we can inspect which correspondences are activated, we can describe the analogy. Beyond saying two things are alike, correspondences encode precisely *how* the two things are alike; with strength, we can justify how much they are alike. Thus, a description of how and why an analogy was made could be explicated to the user: to represent summation, consider stacking the dots horizontally or vertically.

3.2 Property probabilities

If we consider words to be the atomic units of written English, we can compute an occurrence probability derived from its frequency in a particular corpus. Similarly for representations, the property is the atomic unit; each has an occurrence probability derived from its frequency in a particular corpus.[2] Under a Bayesian interpretation, we can assign each property a prior probability. Thus each property p in a representation is associated with a probability $\Pr(p)$.

While having a baseline probability for individual properties is useful, context gives us further clues. For two analogous problems in different representations, knowing which properties are present in one will update our knowledge of the properties present in other. Returning to our example of adding integers, observing the $\sum$ operator in Equation (1) primes us to expect stacking—whether horizontal or vertical—in Figure 1. The conditional probability $\Pr(p_2 \mid p_1)$ expresses our knowledge about p_2 after observing that p_1 is present.

We define $\Pr(p_2 \mid p_1)$ as per the usual definition, adapted to our domain and corpus. For a representation A, we can write problems a_i. Each problem can potentially be transformed into suitable problems $T_B(a_i)$ in representation B. Note that $T_B(a_i)$ is a set; more than one transformation might be appropriate. Using the count operator #, and $\mathrm{sat}(p, q)$ as a predicate on whether the property formula p is satisfied by the properties in question q, we have

$$\Pr(p_2 \mid p_1) = \frac{\sum_i \# \left\{ b_j \in T_B(a_i) \mid \mathrm{sat}(p_1, a_i) \wedge \mathrm{sat}(p_2, b_j) \right\}}{\sum_i \# \left\{ b_j \in T_B(a_i) \mid \mathrm{sat}(p_1, a_i) \right\}}. \quad (5)$$

Transformations can be imperfect; we consider the transformation appropriate if a human expert would be able to reach the same solution as under the original problem statement.

3.3 Strength

Correspondence strength must reflect the analogical similarity of the constituent property formulae. Naively, this is the conditional probability, but conditional probabilities are not directly comparable: Is there a big difference to the prior probability? How much *could* the probabilities be different?

[2]For now we assign prior probabilities based on expert knowledge. In future work we will compute such occurrence probabilities from large corpora of problems and representations.

To address these concerns, we define the strength of the correspondence $\langle p_1, p_2, s \rangle$ to be

$$s = \frac{\Pr(p_2 \mid p_1) - \Pr(p_2)}{1 - \Pr(p_2)}. \quad (6)$$

When $\Pr(p_2 \mid p_1) < \Pr(p_2)$, we redefine the correspondence to be between p_1 and $\textsc{not}\, p_2$:

$$\frac{\Pr(\textsc{not}\, p_2 \mid p_1) - \Pr(\textsc{not}\, p_2)}{1 - \Pr(\textsc{not}\, p_2)} = \frac{\Pr(p_2) - \Pr(p_2 \mid p_1)}{\Pr(p_2)}. \quad (7)$$

Both are undefined when $\Pr(p_1)$ or $\Pr(p_2)$ are 0 or 1. We consider this an acceptable compromise, as this indicates either an 'impossible' property, or a 'guaranteed' property. Neither case is useful in our framework.

Informally, Equation (6) states the increase in probability of observing p_2 relative to the maximum potential increase of observing p_2. The numerator is the difference between the informed and uninformed probability of p_2, while the denominator is the difference between perfect knowledge of p_2 and the uninformed probability of p_2. This balances the need to measure the difference in probability against the confounding effect of p_2 already having a high probability. Similarly, Equation (7) informally states the *decrease* in probability of p_2 relative to the potential decrease of p_2.

Correspondence strength is related to two existing concepts: mutual information [3], and Kullback-Leibler (KL) divergence [9]. Mutual information is a symmetric measure of how much information is shared between two distributions. This symmetry makes it inappropriate for our use-case: correspondences are not necessarily symmetric, one 'direction' can be stronger than the other. KL divergence is asymmetric, but not bound to the interval $[0, 1]$. Because properties behave as Bernoulli random variables, we can normalise the KL divergence to the interval $[0, 1]$ with information content: $\mathrm{KL}(\Pr(p_1 \mid p_2) \parallel \Pr(p_1))/\mathrm{I}(p_1)$. But KL divergence forms a leaky abstraction. As we will see in Section 4, strength as defined in Equation (6) encapsulates relationships between property formulae only in terms of prior probabilities and strength; KL divergence requires calculations on posterior probabilities that are not necessarily available.

4 EXPLANATIONS

Correspondences are central to our representation recommendation process, so we need to understand how they are discovered and how they can be interpreted as explanations for the recommendation.

4.1 Discovering correspondences

Consider the correspondence between numbers and dots, specifically, between 2 and ∘∘: these fill the same role in their respective representations. Hence,

$$\langle\ (\text{token}, 2, \{\text{hasType} = \text{number}\}),\ (\text{token}, \circ\circ, \{\text{hasType} = \text{dot-arrangement}\}),\ 1.0\ \rangle$$

Where does this come from? For a human this is 'obvious', but it must be deduced from somewhere. We propose five rules to automatically discover correspondences split into three groups: identity, correspondence-based, and representation-based. What follows is a high-level summary; further technical details are in Appendix A. Figure 3 shows a diagrammatic interpretation of four of the rules.

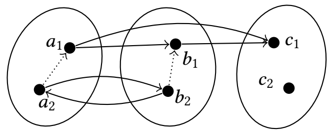

Figure 3: A diagram of three representations, each with two properties. Some properties are related through an attribute, the dashed arrow. Correspondences are solid arrows. Some example discoveries are: a_1b_1 and b_1c_1 generate a_1c_1 with [CMP]; a_1b_1 generates a_2b_2 with rule [VAL]; a_2b_2 generates a_1b_1 with rule [ATR]; and a_2b_2 generates b_2a_2 with rule [REV].

The rule of identity states that two properties with the same kind and same value are perfectly corresponding:

$$\frac{p_1 \equiv p_2}{\langle\, p_1,\ p_2,\ 1.0 \,\rangle}\ [\textsc{idy}] \tag{8}$$

where the relation ≡ is the string match on kinds and values. This rule implies that correspondences are reflexive with strength 1, and allows naturally overlapping representations (e.g., algebra and a tabular representation may reuse numbers) to map cleanly into one another.

Correspondence-based rules build new correspondences from existing ones. The first is the rule of reversal,

$$\frac{\langle\, p_1,\ p_2,\ s \,\rangle}{\langle\, p_2,\ p_1,\ s' \,\rangle}\ [\textsc{rev}] \tag{9}$$

where

$$s' = s \cdot \frac{\Pr(p_1)}{1-\Pr(p_1)} \cdot \frac{1-\Pr(p_2)}{\Pr(p_2)}. \tag{10}$$

This allows us to 'walk backwards' along a correspondence. The second is the rule of composition,

$$\frac{\langle\, p_1,\ p_2,\ s \,\rangle \quad \langle\, p_2,\ p_3,\ s' \,\rangle}{\langle\, p_1,\ p_3,\ s \cdot s' \,\rangle}\ [\textsc{cmp}] \tag{11}$$

allowing us to chain together correspondences. Note that due to independence assumptions, the correspondence between p_1 and p_3 might be stronger than calculated; the product $s \cdot s'$ is a conservative suggestion.

Finally, representation-based rules exploit the internal structure of each representation to suggest new 'parallel' correspondences, as illustrated in Figure 3 by a_1b_1 and a_2b_2. The two rules are dual: the rule of attributes

$$\frac{\langle\, p_1,\ p_2,\ s \,\rangle \quad \mathrm{attr}(p_1, l, e_1) \quad \mathrm{attr}(p_2, l, e_2)}{\langle\, e_1,\ e_2,\ s \,\rangle}\ [\textsc{atr}] \tag{12}$$

and the rule of values

$$\frac{\langle\, e_1,\ e_2,\ s \,\rangle \quad \mathrm{attr}(p_1, l, e_1) \quad \mathrm{attr}(p_2, l, e_2)}{\langle\, p_1,\ p_2,\ s \,\rangle}\ [\textsc{val}] \tag{13}$$

where $\mathrm{attr}(p, l, e)$ is a predicate asserting that property p has an attribute with label l and entry e. These rules allow us to use an 'internal relationship' (encoded with attributes) such as type information to build inter-representational correspondences.

Let us explore these rules in action using our examples of algebra and dots. Table 1 lists a subset of the properties from each representation.[3]

Table 1: Some properties of the algebraic and dot representations. We will use these to build correspondences.

Repr.	Properties
Algebraic	(type, number, ∅)
	(token, 1, {hasType = number})
Dot	(type, dot-arrangement, ∅)
	(token, ∘, {hasType = dot-arrangement})

Because these two representations are disjoint—no properties are equivalent, sharing a kind and value—the rule of identity [IDY] cannot be used. We must provide an initial seed correspondence, something the analyst might notice quickly. For this, we use

$$\langle\ (\text{token}, 1, \{\text{hasType} = \text{number}\}),\ (\text{token}, \circ, \{\text{hasType} = \text{dot-arrangement}\}),\ 1.0\ \rangle$$

That is, 1 corresponds perfectly to ∘. From this seed we can apply rules to generate new correspondences. First, we apply the rule of reversal to associate ∘ with 1:

$$\frac{\langle\ (\text{token}, 1, \{\text{hasType} = \text{number}\}),\ (\text{token}, \circ, \{\text{hasType} = \text{dot-arrangement}\}),\ 1.0\ \rangle}{\langle\ (\text{token}, \circ, \{\text{hasType} = \text{dot-arrangement}\}),\ (\text{token}, 1, \{\text{hasType} = \text{number}\}),\ 0.9\ \rangle}\ [\textsc{rev}]$$

Notice the strength reduced slightly: this is because in the catalogue a 1 occurs more often than a ∘. (In this case, ∘∘ is not two ∘s, they are different dot arrangements.)

Using the correspondence between ∘ and 1, we can discover correspondences between their attributes. Applying the rule of attributes, we have a correspondence between numbers and dot arrangements. The derivation is shown in Equation (14) on the following page, where we abuse the attribute notation to state the attributes share a label.

We can continue applying rules in this way to discover new possible correspondences. A richer set of properties would allow for more rule applications and more discoveries.

4.2 Correspondences as explanations

Understanding both the definition and source of correspondences, we can interpret their function in making a recommendation. Correspondences provide two modes of explanation: descriptive explanation, and constructive explanation.

A descriptive explanation is useful when two structures are considered the same; for example, our statements about summing numbers in Equation (1) and arranging dots in Figure 1 are analogous. *How* they are analogous might be unclear, but the correspondences that link them form an explanation. Consider our correspondence

$$\langle\, (\text{token}, 1, a_1),\ (\text{token}, \circ, a_2),\ 1.0 \,\rangle$$

linking 1 and ∘. Or consider the more sophisticated correspondence from Equation (4) linking the $\sum$ operator with stacking. By informing the user that these are the strongest correspondences which are satisfied by both the source and target statements, we explain how they are analogous.

[3]Taking this table as the universe of properties, there are eight potential correspondences, four of which are meaningful. Here we provide one, derive two, and leave the final as an exercise for the reader; there are at least two different derivations of the final correspondence.

$$\frac{\langle\,(\text{token}, \circ, \{\text{hasType} = \text{dot-arrangement}\}),\ (\text{token}, 1, \{\text{hasType} = \text{number}\}),\ 1.0\,\rangle \quad \text{hasType} = \text{hasType}}{\langle\,(\text{type}, \text{number}, \varnothing),\ (\text{type}, \text{dot-arrangement}, \varnothing),\ 1.0\,\rangle}\ [\textsc{atr}] \qquad (14)$$

Conversely we can consider constructive explanations. If a target structure is not known, we can use correspondences to describe what properties the analogous statement should have. Consider again Equation (1), our algebraic problem statement, but this time without knowing the equivalent statement in dots. Using the same correspondences from our descriptive explanation, we can suggest to a student that this problem might be solved by representing 1 as $\circ$, and to use either vertical or horizontal stacking to convey the $\sum$. Such hints can support progress through the problem, and potentially reveal to the student deep insights into numbers and summation.

Contrast our correspondences with alternative approaches: logical or statistical recommendations [1]. Logical approaches are restrictive, requiring strong guarantees about the domains while failing to capture the inherently fuzzy nature of connections people make between domains. In particular, analogies which are merely 'good enough'—in the sense that they have the shape of similarity without being rigorously correct—cannot be reasoned with formally. Statistical approaches solve the problems associated with logical approaches, but obscure the underlying reasoning. Without any internal structure, the suggestions are opaque to the user: there is no justification or support on why this is analogous, and thus a good recommendation. Correspondences aim to allow a degree of uncertainty through strength while retaining sufficient formality through properties to provide valuable explanations.

$$\begin{gathered}(\text{type}, \text{title}, \varnothing) \quad (\text{type}, \text{character}, \varnothing) \quad (\text{type}, \text{actor}, \varnothing)\\ (\text{token}, \text{Blade Runner}, \{\text{isThe} = \text{title}\})\\ (\text{token}, \text{Harrison Ford}, \{\text{isThe} = \text{actor}\})\\ (\text{token}, \text{Rick Deckard}, \{\text{isThe} = \text{character},\\ \text{playedBy} = \text{Harrison Ford}\})\end{gathered}$$

Figure 4: Example properties for a Films category.

$$\begin{gathered}(\text{type}, \text{title}, \varnothing) \quad (\text{type}, \text{character}, \varnothing) \quad (\text{type}, \text{writer}, \varnothing)\\ (\text{token}, \text{The Caves of Steel}, \{\text{isThe} = \text{title}\})\\ (\text{token}, \text{Isaac Asimov}, \{\text{isThe} = \text{writer}\})\\ (\text{token}, \text{Elijah Baley}, \{\text{isThe} = \text{character}\})\end{gathered}$$

Figure 5: Example properties for a Books category.

5 DISCUSSION

Correspondences are one part of our representation recommendation pipeline. Previous work has shown the recommendations in general are meaningful [13]. But it is worth analytically examining the quality of correspondences in isolation. In this section we consider correspondences and the discovery rules with respect to their theoretical motivation and their generality. A discussion about the theoretical limitations of correspondences is in Appendix B.

5.1 Cognitive grounding

Human reasoning takes many forms, but broadly we can describe reasoning as *deductive*, *inductive*, or *abductive* [10]. The simplest to automate—captured by the identity, reversal, and transitivity rules—is deductive reasoning. Through operations on existing correspondences we can deduce new correspondences. Thus, correspondences form a sort of logic.

While the rules of identity, reversal and transitivity are motivated by deductive reasoning, attribute- and value-correspondence rules are motivated by *inductive* reasoning. Specifically, we assume a meaningful structure must be preserved, and this structure is exposed through relationships. By observing and following these relationships, we can infer new correspondences.

Foundational work by Gentner explored how analogical power is proportional to the relations preserved, compared to the superficial attributes[4] preserved [5]. Indeed, Gentner defines an analogy to be 'a comparison in which relational predicates, but few or no object attributes, can be mapped from base to target' [5]. Our rules address this with the 'hasType' attribute: through the types of patterns or relation-tokens, we can extract these relation mappings. The extended type system accommodates higher-order relations.

Our property framework can be extended beyond typing relations with other attributes, which act as meta-relations; these are automatically used by the existing discovery rules.

5.2 Generality

Correspondences, and the rules to discover them, were developed within our rep2rep representation recommendation framework. But the underlying idea—that analogical similarities across domains are discoverable—abstracts to a more general setting. We now define the necessary components to apply correspondences, and generalise the discovery rules on these components (full details are in Appendix C).

First, we can observe that the rules [REV] and [CMP] are context-independent, so we can leave them alone. Second, we observe that 'properties' can be made opaque; instead we assume only a set of atoms. The [VAL] and [ATR] rules relied on attributes from properties, so must be replaced. Instead, we define a single rule [REL], which is the same except for replacing attr(p, l, e) with $R(p, e)$ for some relation R on atoms p and e. Finally, we keep the equivalence relation $\equiv$ from the [IDY] rule; for each setting define it as a special equivalence relation on the atoms. In this way we leave behind the context of representation selection, and instead have a general correspondence framework for any domain.

[4] *Attribute* here is used in the manner of Gentner [5]; these are not what we define as attributes. All other uses of *attribute* outside direct quotes are according to our definition of attribute.

To demonstrate this generalisation, we can encode the problem of recommending a *product* to a *customer*, rather than a representation to a student. Our sets of atoms are no longer representations, but product categories: books and films, for example. We use the rep2rep property structure to help us manage the encoding, so our atoms remain (kind, value, attributes). Our kinds are limited to types and tokens, but the values are diverse, ranging over names, places and dates. We keep our relations as attributes, but use new labels such as 'isThe', or 'playedBy'. The equivalence relation ≡ is string equality on kinds and values. We give some sample properties in Figures 4 and 5.

In our simple example, we can apply the rule of identity to *title*, and also to *character*, because these two categories overlap. From this we can apply the rule of value using the 'isThe' attribute and *character* correspondence, thus deducing that *Rick Deckard* corresponds to *Elijah Baley*. The derivation is:

$$\begin{aligned}\text{Let } p_1 &= (\text{token}, \text{Rick Deckard}, \{\text{isThe} = \text{character},\\ &\qquad\qquad\qquad \text{playedBy} = \text{Harrison Ford}\})\\ \text{and } p_2 &= (\text{token}, \text{Elijah Baley}, \quad \{\text{isThe} = \text{character}\}) \text{ in}\end{aligned}$$

$$\frac{\langle\, (\text{type}, \text{character}, \varnothing),\ (\text{type}, \text{character}, \varnothing),\ 1.0\,\rangle \quad \text{attr}(p_1, \text{isThe}, \text{character}) \quad \text{attr}(p_2, \text{isThe}, \text{character})}{\langle\, p_1,\ p_2,\ 1.0\,\rangle}\ [\text{VAL}]$$

Over such constrained property sets this is trivial, but exemplifies the application of our framework to a new domain.

6 SUMMARY AND CONCLUSIONS

We introduced and motivated a theory of *correspondences*: relationships between property formulae which 'fill the same role' across representations. Our theory of correspondences is grounded in propositional logic and probability, with rules for extending the set of correspondences in an interactive loop with human analysts. Our analysis demonstrates the cognitive basis of these discovery rules, and the generality of correspondences.

We described correspondences' use in building analogies for problem solving through a simple counting problem, and sketched how the same frameworks could be adapted to a domain such as product recommendation. Further applications include automated and interactive theorem proving, business visualisation tools, diagnostic and reporting systems, and many other domains where clear communication with justification is paramount.

In future work we will explore generalisations of rules to work over formulae, automatic description generation from correspondences, and exploring the philosophical position of correspondences in knowledge representation.

ACKNOWLEDGMENTS

Aaron Stockdill is supported by the Hamilton Cambridge International PhD Scholarship. This work was supported by the EPSRC Grants EP/R030650/1 and EP/R030642/1. We wish to thank the four anonymous reviewers, whose comments helped to improve the presentation of this paper.

REFERENCES

[1] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. 2013. Recommender systems survey. *Knowledge-Based Systems* 46 (2013), 109–132.
[2] Joan Condell, John Wade, Leo Galway, Michael McBride, Padhraig Gormley, Joseph Brennan, and Thiyagesan Somasundram. 2010. Problem solving techniques in cognitive science. *Artificial Intelligence Review* 34, 3 (Oct 2010), 221–234.
[3] Thomas M. Cover and Joy A. Thomas. 2005. *Elements of Information Theory.* John Wiley & Sons, Ltd, Hoboken, NJ, USA.
[4] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. 1989. The structure-mapping engine: Algorithm and examples. *Artificial Intelligence* 41, 1 (1989), 1–63.
[5] Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive Science* 7, 2 (1983), 155–170.
[6] Dedre Gentner. 2002. *Analogy in Scientific Discovery: The Case of Johannes Kepler.* Springer US, Boston, MA, USA, 21–39.
[7] Naomi Goldblum and Shifra Glick. 2001. *The Brain-Shaped Mind: What the Brain Can Tell Us About the Mind.* Cambridge University Press, Cambridge, UK.
[8] Tom Hope, Joel Chan, Aniket Kittur, and Dafna Shahaf. 2017. Accelerating Innovation Through Analogy Mining. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '17).* ACM, New York, NY, USA, 235–243.
[9] S. Kullback and R. A. Leibler. 1951. On Information and Sufficiency. *The Annals of Mathematical Statistics* 22, 1 (1951), 79–86.
[10] Gerhard Minnameier. 2010. The logicality of abduction, deduction, and induction. In *Ideas in action: Proceedings of the applying Peirce conference.* Nordic Pragmatism Network Helsinki, 239–251.
[11] Daniel Moody. 2009. The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. *IEEE Transactions on Software Engineering* 35, 6 (2009), 756–779.
[12] Johanna D Moore. 1993. What Makes Human Explanations Effective?. In *Proceedings of the 15th Annual Conference of the Cognitive Science Society.* Lawrence Elbaum Associates, Hillsdale, NJ, USA, 131–136.
[13] Daniel Raggi, Aaron Stockdill, Mateja Jamnik, Grecia Garcia Garcia, Holly E. A. Sutherland, and Peter C.-H. Cheng. 2019. Inspection and Selection of Representations. In *Intelligent Computer Mathematics*, Cezary Kaliszyk, Edwin Brady, Andrea Kohlhase, and Claudio Sacerdoti Coen (Eds.). Springer International Publishing, Cham, 227–242.
[14] JA Robinson. 1971. Computational logic: the unification algorithm. *Machine Intelligence* 6 (1971), 63–72.
[15] Iris Vessey. 1991. Cognitive Fit: A Theory-Based Analysis of the Graphs Versus Tables Literature. *Decision Sciences* 22, 2 (1991), 219–240.
[16] Rina Zazkis and Peter Liljedahl. 2004. Understanding Primes: The Role of Representation. *Journal for Research in Mathematics Education* 35, 3 (2004), 164–186.

A DISCOVERY RULES

In this paper we introduced five rules to discover new correspondences. Here we build on that brief overview.

A.1 Attribute and value

Attributes are used to suggest new correspondences based on existing correspondences. Looking again at our example, we might state that a 2 from our algebraic representation is like a ∘∘ from our dots representation. That is, we have the correspondence

$$\begin{aligned}\langle\, &(\text{token}, 2, \{\text{hasType} = \text{number}\}),\\ &(\text{token}, \circ\circ, \{\text{hasType} = \text{dot-arrangement}\}),\ 1.0\,\rangle\end{aligned}$$

The two corresponding properties both have entries for the 'hasType' attribute; perhaps those entries themselves correspond? This is the first rule for correspondence discovery:

$$\frac{\langle\, p_1,\ p_2,\ s\,\rangle \quad \text{attr}(p_1, l, e_1) \quad \text{attr}(p_2, l, e_2)}{\langle\, e_1,\ e_2,\ s\,\rangle}\ [\text{ATR}] \qquad (15)$$

where $\text{attr}(p, l, e)$ is a predicate asserting that property p has an attribute with label l and entry e. Similarly we can define the discovery rule over matching attributes:

$$\frac{\langle\, e_1,\ e_2,\ s\,\rangle \quad \text{attr}(p_1, l, e_1) \quad \text{attr}(p_2, l, e_2)}{\langle\, p_1,\ p_2,\ s\,\rangle}\ [\text{VAL}] \qquad (16)$$

which is identical, but with the property (p) and entry (e) correspondences swapped.

By way of example, using our 2/∘∘ correspondence and the [ATR] rule, we can derive a correspondence between number and dot arrangements. The original properties are in correspondence, they both have an attribute with a common label, and so we conclude that the values of those labels also correspond.

These rules, and in particular the [VAL] rule, expose the limitations of this system:[5] such a rule will generate many nonsense results. All numbers have the type 'number', and all dot arrangements have the type 'dot-arrangement', but we do not want the Cartesian product of numbers and dot arrangements as correspondences! While unfortunate, this rule redeems itself when considering more complex types. Consider the binary operator +, a token in the symbolic algebra representation; it has type

$$\text{number} \times \text{number} \longrightarrow \text{number}.$$

Consider also stacking dots, a pattern in the dots representation; it has type

$$\text{dot-arrangement} \times \text{dot-arrangement} \longrightarrow \text{dot-arrangement}.$$

Given the correspondence between numbers and dot arrangements, automatically associating these is a valuable contribution to the correspondence set.[6]

Extending these rules to work on correspondences which contain property formulae is conceptually subtle, but in practice simple. Rather than requiring the antecedent correspondence to be exactly between the properties or attributes under consideration, it is sufficient that there be an *implication* relationship between the property formulae in the correspondence, and the properties or attributes in the representation: from property to correspondence on the left, and from correspondence to property on the right. For example, we might use correspondences that contain the OR and AND connectives in the [ATR] rule:

$$\frac{\langle\, p_1 \text{ OR } p_X,\ p_2 \text{ AND } p_Y,\ s\,\rangle \quad \text{attr}(p_1, l, e_1) \quad \text{attr}(p_2, l, e_2)}{\langle\, e_1,\ e_2,\ s\,\rangle}$$

More generally, if we have a property p_a, and a correspondence containing the left property formula p'_a, we can still use the [ATR] and [VAL] rules if and only if $p_a \longrightarrow p'_a$. Conversely, if we have a property p_b, and a correspondence containing the right property formula p'_b, we need $p'_b \longrightarrow p_b$.

A.2 Reversal and composition

Representation tables are not the only source of information that we can use to suggest correspondences. The set of correspondences itself can be used to discover correspondences through two rules: reversal, and composition.

A reversed correspondence is exactly as expected: if we know p_1 corresponds to p_2, then we can be infer that p_2 corresponds to p_1 as well.

$$\frac{\langle\, p_1,\ p_2,\ s\,\rangle}{\langle\, p_2,\ p_1,\ s'\,\rangle}\ [\text{REV}] \tag{17}$$

Importantly, s does not necessarily equal s': in one direction, the correspondence might be strong, while the reversed direction might be weaker. This is analogous to Bayes' Theorem: if it is raining, I can be very confident the grass is wet; if the grass is wet, I might suspect it is raining but cannot be sure.

To determine the reversed strength s', we return to the definition of correspondence strength, Equation (6). We can show that

$$s' = s \cdot \frac{\Pr(p_1)}{1 - \Pr(p_1)} \cdot \frac{1 - \Pr(p_2)}{\Pr(p_2)}. \tag{18}$$

That is, the reversed correspondence strength s' is the original correspondence strength s modulated by the ratio between the probability of each property formula being satisfied or not. Our assumption that $\Pr(p_2 \mid p_1) \geq \Pr(p_2)$ guarantees s' will be between 0 and 1.

Similarly to reversing correspondences, we can compose correspondences. If we have that p_1 corresponds to p_2, and p_2 corresponds to p_3, then we can conclude that p_1 in some way corresponds to p_3.

$$\frac{\langle\, p_1,\ p_2,\ s\,\rangle \quad \langle\, p_2,\ p_3,\ s'\,\rangle}{\langle\, p_1,\ p_3,\ s \cdot s'\,\rangle}\ [\text{CMP}] \tag{19}$$

Consequently the strength of composed correspondences is monotonically decreasing; longer chains result in weaker correspondences. This is a conservative estimate which results from the independence assumption: p_1 and p_3 might be more strongly corresponding, but this is expert knowledge which the analyst must provide.

While the reversal rule generalises to formulae trivially, the composition rule is more subtle. Specifically, we can move from requiring exact equality on property p_2 to requiring only implication. That is, we can rewrite the composition rule as

$$\frac{\langle\, p_1,\ p_2,\ s\,\rangle \quad p_2 \longrightarrow p'_2 \quad \langle\, p'_2,\ p_3,\ s'\,\rangle}{\langle\, p_1,\ p_3,\ s \cdot s'\,\rangle}.$$

Knowing more about the properties and the representations these formulae are acting over would allow even greater control: we could move any unsatisfied properties from p'_2 to p_1, for example. But without careful representation checks this would build representation formulae which are heterogeneous—they could only be satisfied by a blended representation. This is beyond the scope of the project.

A.3 Identity

The final correspondence discovery rule is *identity*:

$$\frac{p_1 \equiv p_2}{\langle\, p_1,\ p_2,\ 1.0\,\rangle}\ [\text{IDY}] \tag{20}$$

where $p \equiv p'$ is true if and only if p and p have identical kinds and values. This rule of identity links representations that overlap; we know that this is a correspondence between the two representations. This can help bootstrap the other discovery rules: if properties have the same kind and value, but their attributes are different, then we can apply the attribute rule, [ATR], for example.

A.4 Order and strength

These rules are applied repeatedly until either the analyst is satisfied, or there are no more suggestions. But the rules have no inherent

[5]We will explore this further in Appendix B.
[6]We have extended the rules in (15) and (16) to use unification [14] (treating all types as unique type variables) for the Hindley-Milner–style types.

order, nor are correspondence derivations unique; the same correspondence can be discovered multiple times, and with different strengths. We leave it to the analyst to decide which strength is appropriate, but if they do decide to update the strength we have a problem: any correspondence previously derived from the updated correspondence will in turn need its strength updated. This need propagates, carrying updated strengths through the correspondence set.

To address this, we maintain a dependency graph of derived correspondences to track which correspondences to update. We also use this graph to avoid cycles—ensuring a correspondence is not used to derive itself.

B INCOMPLETENESS AND UNSOUNDNESS

Viewed as an analyst-support tool, our implementation of correspondence discovery is 'complete', or more accurately saturating. If there exists a correspondence which can be discovered by these rules, then our utility will suggest it to the analyst. The tool uses an unpruned graph search over states of the correspondence set, with edges defined by applying the rules to the correspondence set.

When considering the completeness and soundness of the rules, we consider the broader space of valid correspondences. No conditions are both necessary and sufficient for a valid correspondence—such conditions would constitute general knowledge—so our rules will violate one or both of completeness and soundness. Although *both* are violated, we have erred towards avoiding generating a lot of unsound correspondences. This trade-off avoids overwhelming the analyst with many meaningless correspondences, at the cost of potentially missing rare-but-insightful correspondences.

Which correspondences get discovered (completeness) depends on the starting set of correspondences. Assuming an empty set with totally disjoint representations—that is, there are no properties with identical kinds and values—the search is immediately terminated. Even beyond this degenerate case, correspondences which are 'isolated' are still unreachable.

Reaching these unreachable correspondences in a principled way is difficult. Assuming our descriptions of representations are complete, we can generate the countably infinite stream of correspondences; then all such isolated correspondences are reachable. But the correspondences generated will be mostly meaningless, leaving the analyst in no better position than when they started. Thus we will not attempt completeness until stricter correspondence validity conditions are discovered.

Conversely, we are unable to eliminate incorrect (unsound) correspondences from our suggestions. This is primarily an issue of *meaning*: the tokens and types in a representation are given symbols that are meaningful to an analyst. When two symbols occur in the same relations, they are effectively indistinguishable from synonyms. Using types for discovery presents an obvious example: both 2 and 76 have the same type, number, so when presented with ∘∘—of type dot arrangement, known to correspond to number—do we create a correspondence with 2, 76, both, or neither? For such simple examples, domain-specific knowledge can act as a heuristic, but more generally we hit the 'general knowledge' problem: how do we know, a priori, which correspondence is more meaningful?

C GENERALISED RULES

In Section 5, we briefly discussed how the correspondence framework generalises to domains outside problem solving.

Thus the overall structure required for correspondences is $(\mathcal{S}, \Pr(\cdot \mid \cdot), \equiv, \sim)$. Set $\mathcal{S}$ consists of triples $S = (A_S, R_S, \Pr_S)$. Let A_S be a set of atoms, R_S a set of relations on $A_S^2, \ldots, A_S^k$, and $\Pr_S : A_S \to [0,1]$ be a probability function. Further, define a function $\Pr(\cdot \mid \cdot) : A_S \times A_T \to [0,1]$, where the subscripts S and T refer to tuples in $\mathcal{S}$. Similarly we define an equivalence relation $\equiv : A_S \times A_T$. We also require an equivalence relation $\sim : R_S \times R_T$ between the relations of each triple.

With this generalised structure, we can update our discovery rules for correspondences.[7] Reversal and transitivity remain unchanged, as they are independent of the structure. Identity is simple:

$$\frac{a \equiv b}{\langle\, a,\, b,\, 1.0 \,\rangle}\ [\textsc{idy}] \tag{21}$$

So long as a and b are equivalent, then we can put them in correspondence automatically. The equivalence relation should be simple; a typical implementation would be equality.

The two remaining discovery rules—for attributes, and for values—collapse down to a single rule. We define the rule on relations as

$$\frac{\begin{array}{c} r \sim r' \quad r(a_1, \ldots, a_n) \quad r'(b_1, \ldots, b_n) \\ \langle\, a_1,\, b_1,\, s_1 \,\rangle \quad \cdots \quad \langle\, a_k,\, b_k,\, s_k \,\rangle \end{array}}{\langle\, a_{k+1},\, b_{k+1},\, s' \,\rangle \quad \cdots \quad \langle\, a_n,\, b_n,\, s' \,\rangle}\ [\textsc{rel}] \tag{22}$$

where we compute s' by

$$s' = \frac{1}{n-1} \sum_{i=1}^{k} s_i. \tag{23}$$

This rule states that if the equivalent predicate is satisfied by some number of corresponding atoms, then the remaining atoms may also correspond. We only consider the case where $k \geq 1$.

Note in s' the difference between the sum limit k and the fraction denominator $n-1$. This reflects the proportion of the predicate already satisfied: if the predicate is almost complete, the strength should be closer to that of the antecedent correspondences; if the predicate is mostly inferred, the strength should be appropriately weaker. The normalising term $n-1$ occurs because we fix $k \geq 1$, leaving at most $n-1$ degrees of freedom. In the binary case, this is a copy of the antecedent correspondence strength, $s' = s$.

Much like in the special case of attributes and values, we do not need a literal equality between the atoms in the relation and the atoms in the correspondences. We need only that for atom a_i in the relation r and property formula a_i' in the left of the correspondence that $a_i \to a_i'$, and for the atom b_i in the relation r' and property formula b_i' in the right of the correspondence that $b_i' \to b_i$.

[7]In the following rules, we abstract away the matching on $S, T \in \mathcal{S}$, etc. We assume that $a \in A_S$, $b \in A_T$, $r \in R_S$, and $r' \in R_T$.