Inspection and selection of representations*

Daniel Raggi[1], Aaron Stockdill[1], Mateja Jamnik[1],
Grecia Garcia Garcia[2], Holly E. A. Sutherland[2], and Peter C.-H. Cheng[2]

[1] University of Cambridge, UK
{daniel.raggi, aaron.stockdill, mateja.jamnik}@cl.cam.ac.uk
[2] University of Sussex, UK
{g.garcia-garcia, h.sutherland, p.c.h.cheng}@sussex.ac.uk

Abstract. We present a novel framework for inspecting representations and encoding their formal properties. This enables us to assess and compare the informational and cognitive value of different representations for reasoning. The purpose of our framework is to automate the process of representation selection, taking into account the candidate representation's match to the problem at hand and to the user's specific cognitive profile. This requires a language for talking *about* representations, and methods for analysing their relative advantages. This foundational work is first to devise a computational end-to-end framework where problems, representations, and user's profiles can be described and analysed. As AI systems become ubiquitous, it is important for them to be more compatible with human reasoning, and our framework enables just that.

Keywords: Representation in reasoning · heterogeneous reasoning · representation selection · representational system.

1 Introduction

The aim of this work is to contribute to the development of AI systems which, similarly to human experts, can pick effective representations for the task at hand.

The effectiveness of a representation depends on its purpose. One kind of representation may be more useful for problem solving, while another may facilitate learning, and some may be more suitable for school children, while another may be useful for the working professional. Thus, for a system to select representations intelligently, it needs to take into account both the *formal* (structure and information) and *cognitive* (user and task) aspects of representations.

This is interdisciplinary foundational work bringing together artificial intelligence, computer science and cognitive science to devise a framework (the language and methods) for analysing and encoding the properties of representations. In this paper, we focus on the formal properties of representations, and the analysis involved in finding a matching representation for the given *problem* amongst a variety of candidate *representational systems*. This analysis is done in

* This work was supported by the EPSRC Grants EP/R030650/1 and EP/R030642/1.

a purely structural and informational manner (i.e., without taking into account the user and task). As we will argue, the formal properties are a foundational layer upon which the cognitive properties of representations depend. The details of the analysis and encoding of cognitive properties, which bring the *user* profile into the picture, are work in progress.

Furthermore, we present a proof-of-concept application of our framework, where multiple representational systems are automatically evaluated relative to a *problem*, by a measure that estimates how likely each representational system is to contain all the ingredients for finding a solution.

Automating the process of analysing and evaluating representations would lead to a new generation of interactive computer systems which adapt intelligently to the user. The automatic selection of effective representations could have applications ranging from intelligent tutoring systems, to systems that aid the working scientist, to fully automated problem-solvers.

2 The role of representation in reasoning

The advantages of particular representations over others have been extensively discussed [25, 6, 5]. Furthermore, there are evident cognitive benefits of multiple representations over single representations [1].

Various formalisations of specific reasoning systems using a single kind of representation have been implemented, including first order [16, 15], higher order [11, 21], diagrammatic [27, 13, 28], among many others. A few heterogeneous reasoning systems that integrate multiple representations have also been built [2, 26], as well as some tools for re-representing problems and knowledge across and within systems [19, 12, 23]. Many systems designed for numerical, algebraic, and geometric computing include tools for representing data in various ways (graphs, plots, figures, diagrams, etc.) [18, 22, 24, 14].

Moreover, in the rapidly-advancing areas of AI (i.e., machine learning), the role that representation plays has been recognised as crucial to the effective processing of data, and *representation learning* has become a rich area of research [3]. However, the gap between the actual computations and the user's understanding seems to be increasing as the tools perform more effectively under more autonomous (and obscure) conditions. To reduce this gap, it is necessary to understand what makes representations better or worse for humans.

Some work has been done to understand the qualities of representations [4, 7]. However, to our knowledge, there is no integration of this knowledge into a framework where representational systems can be analysed, evaluated, and selected computationally, where the task and user can be taken into account. In this work we set some foundations to approach the automation of representation analysis and selection.

2.1 An example

To illustrate the variety and efficacy of representations for reasoning, we first present a problem in probability with three example solutions.

Problem (Birds)**.** *One quarter of all animals are birds. Two thirds of all birds can fly. Half of all flying animals are birds. Birds have feathers. If X is an animal, what is the probability that it's not a bird and it cannot fly?*

Solution (Bayesian)**:** Let the sample space be the set of animals. Let b represent birds, f flying animals, and Pr the probability function. Then, the problem can be reformulated, in the language of conditional probability, as follows:

$$\begin{aligned} &\textit{Assume:} \quad \Pr(b) = \tfrac{1}{4},\ \ \Pr(f|b) = \tfrac{2}{3},\ \ \Pr(b|f) = \tfrac{1}{2}. \\ &\textit{Calculate:}\ \ \Pr(\bar{b} \cap \bar{f}) \end{aligned}$$

To solve this, we start by noting the following facts:

$$\Pr(\bar{b}) = \Pr(\bar{b} \cap \bar{f}) + \Pr(\bar{b} \cap f) \tag{1}$$

$$\Pr(f) = \Pr(b \cap f) + \Pr(\bar{b} \cap f) \tag{2}$$

$$\Pr(\bar{b} \cap f) = \Pr(\bar{b}|f) \Pr(f) = \tfrac{1}{2} \Pr(f). \tag{3}$$

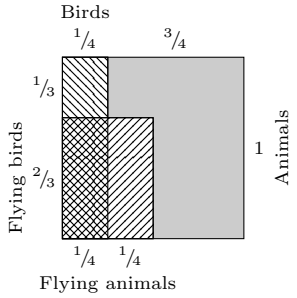
From (2) and (3) we can show that $\Pr(\bar{b} \cap f) = \frac{1}{2} \Pr(b \cap f) + \frac{1}{2} \Pr(\bar{b} \cap f)$, from which we obtain

$$\Pr(\bar{b} \cap f) = \Pr(b \cap f). \tag{4}$$

Thus, we have the following:

$$\begin{aligned} \Pr(\bar{b} \cap \bar{f}) &= \Pr(\bar{b}) - \Pr(\bar{b} \cap f) && \text{from (1)} \\ &= \Pr(\bar{b}) - \Pr(b \cap f) && \text{from (4)} \\ &= (1 - \Pr(b)) - \Pr(f|b) \Pr(b) && \text{from probability axioms} \\ &= \tfrac{3}{4} - \left(\tfrac{2}{3}\right)\left(\tfrac{1}{4}\right) = \tfrac{7}{12}. && \text{from assumptions} \end{aligned}$$ □

Solution (Geometric)**:** In the figure below, the unit square represents the set of animals. The regions representing birds and flying animals are drawn according to the assumptions.



It is clear that the shaded region (non-flying, non-bird animals) has an area of $\frac{3}{4} - \left(\frac{2}{3}\right)\left(\frac{1}{4}\right) = \frac{7}{12}$. □

Solution (Contingency)**:** We start by building a table from the two relevant classes (birds and flying animals) and fill in the values that we are given. Next, the table is filled in gradually, according to simple tabular constraints.

	birds	non-birds	total
flying	(2/3)(1/4)	(2/3)(1/4)	
non-flying		$3/4 - (2/3)(1/4)$	
total	1/4	3/4	1

The rules used to fill the rest of the table are either transferred from the problem's statement (e.g., *half of all flying animals are birds* means that top cells must have the same values), or from probability theorems (e.g., *the total probability is the sum of the parts* means that the total of each column/row must be the sum of the cells in the corresponding column/row). It's worth noting that to the experienced user of contingency tables, the constraints corresponding to probability theorems do not need to be explicitly invoked from probability theory, but simply recalled as tabular/arithmetic manipulations. Thus, we reach the solution when we fill in the shaded cell. □

Note that there may be many solutions using the same representational system (e.g., many Bayesian solutions, many Geometric solutions, etc.). In the next section we present the framework for analysing these representations.

3 Representing representations

Representational systems are diverse, and their designs and specifications are heterogeneous. To do a comparative analysis, we need a common framework that captures the properties that make up each representation.

Our framework is based around three concepts: *representational system* (RS), *problem* (Q), and *correspondence*. In the following subsections we suggest a concrete method for inspecting and encoding their properties. The result is a set of machine-readable *tables* suitable for further analysis.

Table construction currently needs to be done by a human *analyst*, but the purpose of such tables is that they can be processed automatically to yield a representation recommendation.

3.1 Representational system

We view representational systems as consisting of two layers implemented over a medium: a *grammatical* layer, and an *inferential* layer. Broadly speaking, the grammatical layer distinguishes the sensible from the nonsensical, and the inferential layer distinguishes the valid from the invalid.

The table of an RS is a structured description of the RS. It is a collection of the most relevant *properties* of the RS. We encode these properties as pairs

(k, v), where k is the property *kind* and v is the *value*. For example, if an RS table has the entry $(\text{rigorous}, \text{TRUE})$ this means that the RS has the property of being rigorous; and if it has $(\text{token}, +)$ this means that the token '+' is part of the symbols used in the RS (for examples of RS tables, see Tables 1 and 2).

Grammar. The building blocks of an RS are called *tokens*. These are used to construct *expressions*. The *grammar* of an RS allows us to distinguish between admissible and inadmissible expressions. Although grammars can be constructed in various ways (e.g., by production rules [8], or by some underlying type theory [10]), our framework is agnostic to the specific *construction*, but embraces *type theory* as a generic descriptive tool. Thus, every token, and every admissible expression will be assumed to have a *type*. However, to keep our framework flexible, we make no other assumptions about the specific type theory that we use.

We encode the property of *grammatical complexity* (*g-complexity* in Tables 1 and 2) using Chomsky hierarchies, which are linked to well-known classes of computational complexity and expressiveness [9].

Inference. The medium on which an RS is implemented (e.g., a pen and paper, or a computer interface) can be manipulated by some agent. Thus, we need a concept of *state*. Broadly speaking, a state refers to the conditions of a medium at some moment in time. Manipulations are changes of state.

The *inferential* layer of an RS determines how the grammatical expressions of an RS can be manipulated *validly*. Borrowing from the field of computer-assisted theorem proving, we refer to valid manipulations as *tactics*. Moreover, tactics are often parametric on knowledge. The knowledge encoded in an RS can be represented as a collection of *facts*. These can be formulae (e.g., in Bayesian system $\Pr(x) = \Pr(x \wedge y) + \Pr(x \wedge \neg y)$), extrinsically-imposed constraints (e.g., in contingency tables the last value of a row must be the sum of the values of the row), or intrinsic constraints (e.g., the area of a region is the sum of the areas of its parts). Thus, tactics and facts are the main constituent properties of an RS.

As with grammatical complexity, inferential complexity (*i-complexity* in Tables 1 and 2) can be measured according to known standards. The partial order induced by injective embeddings of theories within one another could be a basis by which to compare systems, but a radical simplification by flattening into 5 classes turns out to be enough for a rough assessment of complexity. We propose the following levels (with a known system for reference): 1 (propositional logic), 2 (decidable fractions of arithmetic/set theory), 3 (Peano arithmetic), 4 (constructive set theory), and 5 (Zermelo-Fraenkel).

Furthermore, we include the property of *rigour* to describe whether the calculations/proofs are guaranteed to be exact/correct.

We described the most significant properties of RS tables. Our framework includes more, but it is nevertheless a fixed set of *kinds* for every RS table. This provides a template that can aid an analyst to generate tables, ensures

Table 1. A section of the Bayesian RS table.

kind	**value**
types	real, event
tokens	Ω, $\emptyset$, 0, 1, $=$, $+$, $-$, $*$, $\div$, $\cup$, $\cap$, $\setminus$, $^-$, Pr, $\mid$
g-complexity	type-2
facts	Bayes' theorem, law of total probability, non-negative probability, unit-measure, sigma-additivity, ...
tactics	rewrite, arithmetic calculation
i-complexity	3
rigorous	TRUE

Table 2. A section of the Geometric RS table.

kind	**value**
types	point, segment, region, real, string
tokens	$point, $segment (*the prefix $ denotes a label for an actually pictorial symbol*)
g-complexity	type-3
facts	scale-independence of ratio, non-negative area, area additivity,...
tactics	draw point, draw segment, delete, join, compare sizes
i-complexity	2
rigorous	TRUE

consistency across different RSs, and is sufficient for an in-depth analysis of representations. See Tables 1 and 2 for some example RS tables.

3.2 Problems

Abstractly, a *problem* q is a triple (O_q, G_q, C_q) where O_q is an initial condition, G_q is a goal condition, and C_q is a set of constraints on the paths leading from the states that satisfy O_q to the states that satisfy G_q [20].

This definition is difficult to square with the fact that problem specifications rarely look like such a triple. Our contention is that this is because most problems are about information recovery within a space determined by conventional information-manipulation rules.

For example, the *Birds* problem above is neither explicit about the goal condition nor about any path constraints, and we can only assume that the initial condition is whichever state things are when the problem is presented. However, a competent problem-solver will find a statement such as *what is the probability ...* informative about the *type* of the answer expected (specifically, *ratio*, *percentage*, or *real number* or any data-type usually used to encode probability values). In other words, the problem-solver is expected to perform type-inference to obtain the answer type. Moreover, the rest of the problem statement will typically provide data which, under conventional interpretations, can be manipulated to recover the answer. Thus, given a problem specification q, the infor-

mation (O_q, G_q, C_q) is hidden in the specification and meant to be inferred by the problem solver.

We call the set of paths that satisfy (O_q, G_q, C_q) the *semantics* of the problem. The relation between a problem specification and its semantics is complex, because it requires, first, an understanding of how the specification relates to the triple (deemed *problem understanding* by [20]), and second, knowledge of the paths that satisfy the triple (*problem solving*).

Q tables. Similar to RS tables, Q tables encode the properties of problems (*Q properties*) as kind–value pairs.

The presentation of a problem requires an RS. We write $q : S$ to denote 'a problem q represented under system S'. Then, a problem's properties can be encoded similar to representational systems (e.g., which types and tokens appear). The only difference is that here we qualify the properties by their semantic contribution. Thus, to build a Q table, an *analyst* is required to estimate the semantic contribution of properties. That is, they need to decide the importance of each property relative to the other properties. This means, in particular, that knowledge of individual solutions influences the content and quality of the table. Naturally, we do not assume full knowledge of the semantics, as this would require complete knowledge of solutions. We only assume partial understanding of the goal conditions, and any estimation of the relevance of different properties can help. Moreover, if the problem is presented in a system S and the analyst only knows a solution involving a different system S' where a fact α (native to S') is used, the analyst can include α in the table. This will ensure that we do not miss important knowledge of a problem/solution when building a Q table.[1]

See Table 3 for a Q table for the *Birds* problem.

Properties such as *error allowed* and *answer type* are at the top of the importance hierarchy (purple/essential) because they inform us directly about the goal condition. Specifically, *answer type* refers to the shape of the data (datatype) expected to appear in any state satisfying the goal conditions, and *error allowed* refers to the rigour expected out of the answer (i.e., how permissive is the goal condition).

Next (blue/instrumental) in the hierarchy are any tokens, types, patterns, facts, or tactics with pivotal roles in the solution(s). These properties are informative about how the paths in the state space which lead to solutions look.

One step below (green/relevant) are things which are clearly informative about the semantics of the problem, but which may also contain noise or just be informative as heuristics. This now includes tokens which may not appear in the problem specification but which may be useful along the way.

The lowest classes are either circumstances of the representation (yellow/circumstantial), or outright noise (red/noise)—that is, tokens that either appear in the specification or are evoked by it,[2] which contain no information about the

[1] As a consequence, Q tables may contain elements of multiple RSs. For reasons mentioned in Section 5, this is discouraged whenever it is avoidable.

[2] Those appearing in this example are taken from a semantic net [29].

Table 3. A Q table for the *Birds* problem with its natural language statement. The colour codes for importance relative to information content: essential (purple), instrumental (blue), relevant (green), circumstantial (yellow), noise (red).

kind	**value**
error allowed	0
answer type	ratio
tokens	probability, and, not
types	ratio, class
patterns	_:ratio of _:class are _:class, probability of _:class and _:class
facts	Bayes' theorem, law of total probability, unit measure, additive inverse, commutativity of + ...
tactics	deduce, calculate
tokens	one, quarter, all, animals, birds, two, thirds, can, fly, half, flying, X, animal, probability, cannot
related tokens	times, divided_by, plus, minus, equals, union, intersection, probability, zero, ...
# of tokens	67
# of distinct tokens	31
tokens	feathers
related tokens	beast, animate, creature, wing, aviate, flock, fowl, dame, carnal, being, fauna, ...

semantics. Notice, for example, that the *Birds* problem contains the statement *birds have feathers* which is not used by any solution. Thus, the token 'feathers' is classified as noise. Any tokens related to the zoological interpretation of the problem are taken as noise. Encoding these explicitly may be useful to understand potential missteps or distracting data in the specification.

Every Q table will have a fixed set of property kinds. This provides a template to generate tables—empirically, these proved sufficient for an effective analysis of candidate representations in relation to the problem.

3.3 Correspondence

We have presented a framework for encoding the properties of problems and representational systems. Now we need some method for assessing the relative value of different RS for representing a given problem. Our approach relies on the notion of *correspondence*, which is a way of relating Q properties with RS properties. It is the fundamental notion that we use for calculating the match of an RS to a problem.

Analogical correspondences. Different RSs are linked to each other through structure-preserving transformations. For example, an *event* can be encoded as a *proposition*, or as a *set*, or as a *region* in the plane. These relations between types form the basis of more complex analogies. For example, the conjunction

of two events corresponds to the intersection of two regions, and the *probability* of an event corresponds to the area of its corresponding region. Furthermore, such transformations also induce the correspondence of facts. For example, if probability corresponds to area, then the *law of total probability* corresponds to *area additivity*. It should be noted that for every property p the *reflexive* correspondence (with strength 1) is by default considered in our framework. The logic and mechanisms for reasoning through such transformations has been explored elsewhere [23]. In this work, rather than the mechanism, we are concerned with assessing the relative value that such transformations provide.

Q-specific correspondences. The analogical correspondences are induced by transformations between RSs (i.e., the mapping between tokens/types/facts/tactics). However, other problem-specific information may also be valuable for assessing RSs. For example, the error allowance of a problem informs us whether we need a rigorous RS or whether an imprecise one is sufficient (if there are other reasons for it to be valued, e.g., an approximate solution is sufficient for young children). Thus, correspondences such as between the Q property *error allowed = 0* and the RS property *rigorous* can be included.

Correspondence tables. Currently, we assume that a catalogue of such transformations/analogies is known. Furthermore, we assume that we have a measure that estimates the information loss in a correspondence. We call this the correspondence *strength*.[3] For example, the injection of natural numbers into real numbers is lossless, so the strength of *type natural* to *type real* is assumed to be 1. However, any transformation from real to natural is lossy, so its strength must be less than 1 (the question of how much exactly is up for discussion). Thus, each correspondence can be encoded as a triple (p_q, p_r, s) where p_q and p_r are Q and RS properties, respectively, and s is the strength.

Each correspondence between a property of q and a property of S can be seen, roughly, as a reason why q could be represented in S. Simplistically, this could mean that adding up the values of all correspondences between q and S might give us a score of S. However, the reasons may not be independent, so adding them up may count redundant correspondences multiple times. Thus, we introduce a simple calculus for specifying correspondence in the most independent possible way (e.g., see Table 4; more details are in Section 4.2).

But how can Q, RS, and correspondence tables be used for representation recommendation? We present a proof-of-concept algorithm next.

4 Using the framework

Our tables give us properties of the problem and of the candidate RSs. Correspondence tables give us explicit links between them. The task now is to exploit

[3] In future work, we will investigate how correspondences and their strength can be identified automatically (e.g., using machine learning).

Table 4. Some example correspondences encoded with operator OR.

Q property formula	RS property	strength
type occurrence OR type class	type event	1
type ratio OR type percentage	type real	1
token intersection OR token and	token ∩	1
token given OR token if	token |	1
error allowed = 0	rigorous	1
error allowed = 0	NOT rigorous	−1

this information to find correspondences which match properties of both the problem and the target RS. Not all properties bear equal importance, thus we modulate the correspondence strength. Combining these assigns a real value to each potential RS indicating its relevance as a candidate RS. Algorithm 1 implements this process.

Algorithm 1 Uses properties of problems and representational systems to rank candidate RSs.

```
LoadTables()
recommendations ← []
for each representation do
  t ← 0    // t is the score
  for each correspondence do
    prop_q ← Properties(problem)
    prop_r ← Properties(representation)
    (p_q, p_r, strength) ← correspondence
    importance_corr ← Max(Importance, p_q)
    if Match(prop_q, p_q) and Match(prop_r, p_r) then
      t ← t + importance_corr × strength
    end if
  end for
  if t > 0 then
    Append(recommendations, ⟨t, representation⟩)
  end if
end for
return Sorted(recommendations)
```

4.1 Matching correspondences

In our running example, we have a Q table for the *Birds* problem and four RS tables for the representational systems: Bayesian, Geometric, Contingency, Natural Language, and also an Euler RS table.[4] They are accompanied by a

[4] By 'Euler' we mean some implementation of Euler diagrams.

table of correspondences between properties. Algorithm 1 accesses these tables, and then iterates over the RSs to find correspondences linking the problem to the RS.

Suppose the first candidate representation for the *Birds* problem is Bayesian. Thus, we consider the Bayesian RS table and the *Birds* problem Q table. Next, we examine each correspondence: a triple (p_q, p_r, s) where p_q and p_r are properties of the problem and representation, respectively, and $s \in [-1, 1]$ is the correspondence strength. We examine if both Q table properties and RS table properties match the conditions of this correspondence. For example, the correspondence:

$$(\text{error allowed} = 0,\ \text{rigorous},\ 1)$$

from Table 4 matches properties in the Q table in Table 3 *and* those in the RS table from Table 1. If there is no match, we disregard this correspondence. When they do match—as in this example—we take the strength s and modulate it by the importance of the matched Q property. Each importance colour band is assigned a value in the $[0, 1]$ interval, which we multiply by the strength of the correspondence s. Our example correspondence involves the property *error allowed*, which is an essential (purple) property, so is modulated by the 'essential' value 1. Altogether, this correspondence contributes $1 \times 1 = 1$ to the Bayesian RS ranking score.

4.2 Property formulae

The p_q and p_r from the correspondence triple can be *property formulae* expressed in a simple calculus using binary connectives AND and OR, and the unary connective NOT.[5] The property calculus allows for greater expressivity and better captures the nature of correspondences between Q and RS properties. We see this in the correspondence:

$$(\text{type occurrence OR type class},\ \text{type event},\ 1)$$

where we require one (or both) of the properties specified in the p_q of the correspondence triple to occur in the Q table. In this situation, we do observe *type class* occurring as an instrumental (blue) property. The correspondence is matched despite the absence of *type occurrence*; notice that it was necessary to observe *type event* in the Bayesian RS table. The matched correspondence formula involved both *type class* and *type occurrence*, with the match being satisfied by an instrumental (blue) property. Thus, the correspondence strength of 1 is modulated by the 'instrumental' value[6] 0.6, increasing the Bayesian RS score to $1 \times 0.6 = 0.6$.

[5] AND requires that both properties appear in the property table. OR requires that at least one of the properties appears in the property table. If both properties appear in the table, the strength is only counted once. NOT requires that a specified property does not occur in the property table.

[6] The value 0.6 for 'instrumental' properties is chosen arbitrarily; the only condition is that the value-importance relation is monotonic. In future work, these parameters should be tuned with experimental data.

4.3 Making a recommendation

Once all correspondences for a particular Q and RS are identified and modulated by importance, we combine them to a single score. This can be done in many ways: we take a simple sum. For the example of the Bayesian representation and the *Birds* problem, the correspondence analysis gives an overall score of 9.3.

Repeating the scoring process above for each candidate RS yields the following recommendation ranking:

Bayesian	9.3
Geometric	7.2
Natural Language	6.9
Contingency	5.4
Euler	1.5

We hence recommend that the *Birds* problem, initially posed in a Natural Language RS, might be better attempted in the Bayesian RS. This seems a sensible recommendation.

5 Discussion and future work

We showed that representation selection can be encoded in a sufficiently formal way to computationally analyse the underlying informationally-relevant structural matches across domains. This is novel and exciting. We now evaluate the quality of performance of our framework, and discuss its significance for applications and future work.

5.1 The influence of known solutions

In this paper we presented an example of an RS recommendation where the input was a Q table (for the *Birds* problem), five candidate RS tables, and the associated correspondence tables. The Q table encoded the problem expressed in natural language, but it contained as *facts* some theorems from other RS, like *Bayes' theorem* in the formal Bayesian solution. We allow the Q table to be heterogeneous (e.g., consisting of properties of more than one RS), and thus include in the Q table the Bayesian facts. Clearly, such properties will boost the score of the Bayesian RS.

More generally, every property in Q table that is native to an RS will boost the score of such an RS. This implies that the RS in which the problem is stated (natural language in our example) will score points just for the fact that the problem is expressed in that RS. However, it might still not get the highest score, because it may contain foreign properties, or properties that clash with the properties of the problem (e.g., the *rigour* property of the RS may clash with the *error allowed* condition of the problem).

Thus, known solutions, regardless of their representation, can influence the recommendation if the analyst introduces their properties in the Q table. Therefore, the analyst must consider:

1. Heterogeneous tables need to be built carefully to avoid redundancy. For example, the *Law of Total Probability* is a Bayesian fact that corresponds to the Geometric fact *Additivity of Areas*. Thus, if both are added to a single Q table, they may result in unjustified boosting of the score. For this reason, the Q tables should be as homogeneous as possible.
2. Some aspects of known solutions do not affect the formal score of an RS. For example, the length of the solution is not considered because, although a shorter solution may be desirable, it is not *informationally relevant*. But it may be relevant from the cognitive point of view (e.g., in the processing cost), and thus forms part of the cognitive properties. Incorporating cognitive properties into our analysis and recommendation framework is part of our ongoing work.

5.2 Analysing the trace

Perhaps more interesting than the resulting scores is the data that the *trace* of the algorithm execution provides. For example, it enables an analysis about how individual correspondences contribute to the total score. The high importance (essential) Q property *answer type ratio* corresponds to properties *type real* in the Bayesian, Geometric, and Contingency systems. However, in the Euler system it has no corresponding property.[7] Similarly, the token *probability* corresponds to the token Pr in Bayesian, and the tactics *compare sizes* and *compare cell values* in Geometric and Contingency, respectively, but it has no correspondence in Euler. Thus, due to the high values for the importance of the essential and instrumental properties, the gap between Euler and the other RSs widens. This is expected and desirable.[8]

We observe that tokens in some systems (e.g., *probability*) can correspond to tactics (e.g., compare sizes) without corresponding to any specific tokens. This is interesting from the cognitive perspective, because these tokens and tactics are very different operationally. A cognitively focused analysis may be able to assess the impact of differences of this sort.

The trace analysis has many potential applications. We envision *tutoring systems* that can make specific recommendations (e.g., "maybe you can draw a region in space to represent the class of animals"), or explainable AI systems that can justify their decisions in a humanly-understandable manner.

Zero-weight properties. Some Q and RS properties make no contribution in terms of correspondence scores, for example, all the *circumstantial* and *noise* properties. Nevertheless, we chose to encode them in our framework because

[7] In Euler diagrams the cardinality of sets is abstracted away; the size of zones is meaningless.

[8] Note that the strength of the correspondences from *probability* to Pr and *compare sizes* was set to 1 (because in principle any probability function is representable in the Bayesian or Geometric systems), but it was set to 0.5 for Contingency because not every probability function is representable in Contingency tables.

they have potentially important effects on cognitive processes. For example, the total number of (circumstantial) tokens may be used to estimate the cognitive cost of registering a problem specification. Moreover, the inference power of an RS may make it more applicable from a formal point of view, but the cost of using it may be higher from a cognitive point of view.

An analysis of the correspondence between noise properties of Q and RS may be used for predicting human error or bias. For example, a novice user might be tempted to represent zoological facts given in the *Birds* problem, but they contribute to unnecessary cognitive costs.

5.3 Evaluating the framework and algorithm

To our knowledge, no work has been done on computationally modelling representation selection, so we have no benchmarks by which to judge our framework or algorithm. The execution of the algorithm presented here is a proof-of-concept application, but it shows that given our encoding of a problem Q and various representations RSs, we can compute interesting measures. Moreover, the approach that we take for encoding the description of problems, RSs and correspondences makes minimal assumptions about them. This makes it general. So far, we have encoded 9 different RSs and various problems.

One of the main limitations of our framework is the need for an analyst to encode the Q and RS properties, the correspondence strengths, and the importance that each Q property has relative to potential solutions. This clearly requires the analyst to understand the complexity of a problem, and to have at least some understanding of how a solution would look (e.g., identifying potentially instrumental facts). This poses a problem for automation. One way of tackling this is with the help of machine learning methods similar to the work of [17] for lemma selection.

5.4 Future work

Our framework opens up many avenues for future research. Automating the generation of Q and RS tables and their importance is a clear goal to be achieved. We are currently including methods for analysing the cognitive properties of representations, and want to extend the framework to include user profiles next. We are curious to find out if representation selection based on our framework can promote problem solving or learning in humans, and want to incorporate it into a personalised multi-representation tutoring system.

6 Conclusion

We have presented a novel framework for computationally selecting suitable representations. We introduced the language, data structures, and methods for encoding and analysing the properties of problems, of representational systems, and the correspondences between them.

Our proof-of-concept algorithm ranks representational systems according to a measure of suitability given a problem. The algorithm analyses the problem's properties in terms of their informational contribution and estimates the likelihood that the problem's semantics can be recovered in each candidate RS. We see this work as an exciting foundation upon which we can build the machinery to analyse cognitive properties, so the user profile may be included to calculate a recommendation.

Acknowledgements

We thank the 3 anonymous reviewers for their comments, which helped to improve the presentation of this paper.

References

1. Shaaron Ainsworth. The functions of multiple representations. *Computers & education*, 33(2-3):131–152, 1999.
2. Dave Barker-Plummer, John Etchemendy, Albert Liu, Michael Murray, and Nik Swoboda. Openproof-a flexible framework for heterogeneous reasoning. In *International Conference on Theory and Application of Diagrams*, pages 347–349. Springer, 2008.
3. Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.
4. Alan Blackwell and Thomas Green. Notational systems–the cognitive dimensions of notations framework. *HCI models, theories, and frameworks: toward an interdisciplinary science. Morgan Kaufmann*, 2003.
5. Peter C-H Cheng. Unlocking conceptual learning in mathematics and science with effective representational systems. *Computers & Education*, 33(2-3):109–130, 1999.
6. Peter C-H Cheng. Probably good diagrams for learning: representational epistemic recodification of probability theory. *Topics in Cognitive Science*, 3(3):475–498, 2011.
7. Peter C-H Cheng. What constitutes an effective representation? In *International Conference on Theory and Application of Diagrams*, pages 17–31. Springer, 2016.
8. Noam Chomsky. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124, 1956.
9. Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.
10. Thierry Coquand. Type theory. *Stanford Encyclopedia of Philosophy*, 2006.
11. John Harrison. HOL light: An overview. In *International Conference on Theorem Proving in Higher Order Logics*, pages 60–66. Springer, 2009.
12. Brian Huffman and Ondřej Kunčar. Lifting and transfer: A modular design for quotients in isabelle/hol. In *International Conference on Certified Programs and Proofs*, pages 131–146. Springer, 2013.
13. Mateja Jamnik, Alan Bundy, and Ian Green. On automating diagrammatic proofs of arithmetic arguments. *Journal of logic, language and information*, 8(3):297–321, 1999.

14. Jupyter. jupyter.org.
15. Matt Kaufmann and J Strother Moore. Acl2: An industrial strength version of nqthm. In *Proceedings of 11th Annual Conference on Computer Assurance. COMPASS'96*, pages 23–34. IEEE, 1996.
16. Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In *International Conference on Computer Aided Verification*, pages 1–35. Springer, 2013.
17. Daniel Kühlwein, Jasmin Christian Blanchette, Cezary Kaliszyk, and Josef Urban. Mash: machine learning for sledgehammer. In *International Conference on Interactive Theorem Proving*, pages 35–50. Springer, 2013.
18. Matlab. mathworks.com.
19. Till Mossakowski, Christian Maeder, and Klaus Lüttich. The heterogeneous tool set, hets. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 519–522. Springer, 2007.
20. Allen Newell. *Human Problem Solving*. Prentice-Hall, Inc., 1972.
21. Tobias Nipkow, Lawrence C Paulson, and Markus Wenzel. *Isabelle/HOL: a proof assistant for higher-order logic*, volume 2283. Springer Science & Business Media, 2002.
22. GNU Octave. octave.org.
23. Daniel Raggi, Alan Bundy, Gudmund Grov, and Alison Pease. Automating change of representation for proofs in discrete mathematics (extended version). *Mathematics in Computer Science*, 10(4):429–457, 2016.
24. SageMath. sagemath.org.
25. Gem Stapleton, Mateja Jamnik, and Atsushi Shimojima. What makes an effective representation of information: a formal account of observational advantages. *Journal of Logic, Language and Information*, 26(2):143–177, 2017.
26. Matej Urbas and Mateja Jamnik. A framework for heterogeneous reasoning in formal and informal domains. In T. Dwyer, H.C. Purchase, and A. Delaney, editors, *Diagrams*, volume 8578 of *Lecture Notes in Computer Science*, pages 277–292. Springer, 2014.
27. Matej Urbas, Mateja Jamnik, Gem Stapleton, and Jean Flower. Speedith: a diagrammatic reasoner for spider diagrams. In *International Conference on Theory and Application of Diagrams*, pages 163–177. Springer, 2012.
28. Daniel Winterstein, Alan Bundy, and Corin Gurr. Dr. doodle: A diagrammatic theorem prover. In *International Joint Conference on Automated Reasoning*, pages 331–335. Springer, 2004.
29. WordNet. wordnet.princeton.edu, 2010.