sciendo FCT Fundação para a Ciência e a Tecnologia MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR CIUHCT Ludus

Next Level Odd-One-Out Puzzles

Benjamin Berger
Leibniz Universität Hannover

Abstract

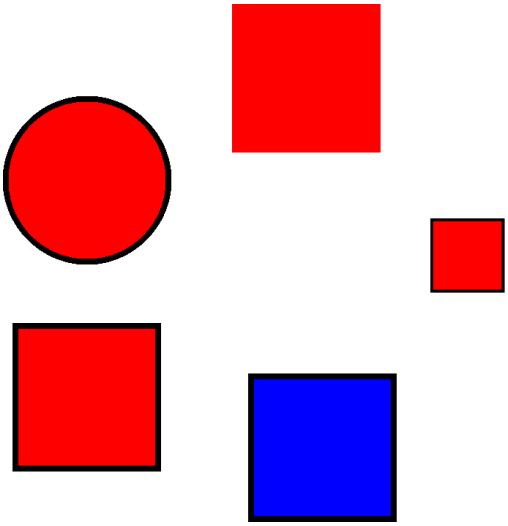
A commonly occurring task in intelligence tests or recreational riddles is to "find the odd one out", that is, to determine a unique element of a set of objects that is somehow special. It is somewhat arbitrary what exactly the relevant feature is that makes one object different. But once that is settled, the answer becomes obvious. Not so with a puzzle popularized by Tanya Khovanova to express her dislike for this type of puzzle. Here, it is a more complicated relation between the objects and the features that determines the odd object, because there is only one object that does not have a unique feature expression. This puzzle inspired me to look for even more complicated relations between objects, features and feature expressions that appear to be even more symmetric, but actually still single out a "special object". This paper provides useful definitions, a theoretical basis, solution algorithms, and several examples for this kind of puzzle.

Keywords: puzzles, symmetry, combinatorics

1 Introduction

A certain puzzle is somewhat popular on the internet which was devised in its present, colorful form by Khovanova [3], but is based on an older greyscale version of the same idea. The task is to identify one of several objects in a picture (essentially reproduced in Figure 1 on the following page) that is "odd". Khovanova states that she does not like these puzzles because of their arbitrariness (a sentiment she shares with Martin Gardner, in one of whose books [2] she saw the original version, which was designed by Tom Ransom [1]), so she (re)invented one. She apparently did so to subvert the concept of odd-one-out puzzles, because the joke is that, while at first glance each of the objects seems to be special in its own way, there is one (the big red bordered square) which does not have a feature expression that is unique. It has nothing special; that is what makes it special. I found this puzzle amusing but too easy and set out to make harder ones, calling them "ooops" (Odd-one-out puzzles).

DOI 10.2478/rmm-2020-0003



A big blue square with black border
A small red square with black border
A big red square without border
A big red circle with black border
A big red square with black border

Figure 1: The puzzle that inspired this paper

2 Problem Description

A puzzle of the kind described in this paper is presented to the human solver as a picture showing several *objects*. Each object has several *features*, such as shape, overall size, border style, color, number of holes, to name a few. Each feature has several *expressions*. For example, "color" may have the expressions "red", "green", or "blue", and "border style" may have the expressions "absent", "solid", or "dashed".

The task is to find out the one object that is "unique" or "special". In interesting puzzles, this is complicated by the fact that there are either no or several objects that have a unique expression of some feature, so one has to consider how the objects relate to each other by having common expressions of some features.

A priori, all objects and all features stand on an equal footing. Also, all expressions of a feature stand on an equal footing and may not be compared to the expressions of other features. There is nothing special about color or dashed lines just because they feel more salient visually, just as there is nothing special about the leftmost object just because it comes first in most peoples' reading habits. This can be expressed more rigorously by stipulating that, whenever the puzzle is changed by exchanging any two features

(possibly requiring new expressions for one feature to be invented if they had a different number of expressions), or by permuting the expressions of some feature, the object that is the solution of the puzzle stays the same, except that of course the feature exchange or expression permutation has been applied to it. As an example for an exchange of features, consider the injective mappings

$$\{\text{red} \mapsto \text{absent}, \text{green} \mapsto \text{solid}, \text{blue} \mapsto \text{dashed}\}$$

and

$$\{\text{absent} \mapsto \text{blue}, \text{solid} \mapsto \text{green}, \text{dashed} \mapsto \text{red}\}$$

These can be used to exchange the features "color" and "border style": for each object, apply the respective mapping to the expression of the respective feature to obtain the expression of the other feature on the transformed object. For example, a borderless green object will become a blue object with solid border. Features that can be exchanged in a way that leaves all objects as they were before (because their expressions correlate perfectly) are not to be considered distinct features.

Also, only expressions of the same feature can be compared among each other, and can only be compared for equality. The justification for a solution is then a statement that is true about only one object (namely the solution) and that treats all features and feature expressions on an equal footing and that is built from only equality comparisons of expressions that belong to the same feature. For example, the justification for why the red big bordered square is the solution for Figure 1 on the preceding page could be given as: "It is the only object so that there is no feature that has a unique expression on that object". Another justification would be "It is the only object that differs from each other object in exactly one feature". However, it is an invalid justification to say "The blue object is special because color is the most salient feature, and it is the only one with that color". Saying this would a priori prefer color, which is not permitted.

The question of what the relevant features and their expressions are is left to the common sense of the solver, but once this step is accomplished successfully, the process becomes purely logical. Therefore I will introduce a mathematical encoding for these concepts in subsection 3.1 on the following page, but may sometimes use the labels "object", "feature" or "feature expression" when talking about the respective mathematical objects.

2.1 Examples

In this subsection, I will give a few example puzzles. To prevent any ambiguity in interpreting the puzzles, Table 1 on the next page lists the features and their possible expressions that were used in all figures except Figure 12. Note that "border width" and "hole size" are not separate features in this paper, since they correlate perfectly with overall size.

Feature	Possible expressions	Remarks
Size	big, small, medium	Size affects border width and hole size
Color	red, green, blue, yellow, white	
Shape	circle, triangle, square, pentagon	
Border style/color	black, dashed black, absent, gray, cyan, green	
Hole count	0, 1, 2, 3, 4, 5	
Hole shape	circle, triangle, square, pentagon	
Hole filling	white, black, red, orange, magenta, cyan, dark green	Never the same as hole border color
Hole border	black, dashed black, absent, gray, white, dark cyan	Same as outer border in Figures 2 and 4

Table 1: List of the features and feature expressions used in the figures. Not every figure uses all features. Not every feature occurring in a figure uses all possible expressions.

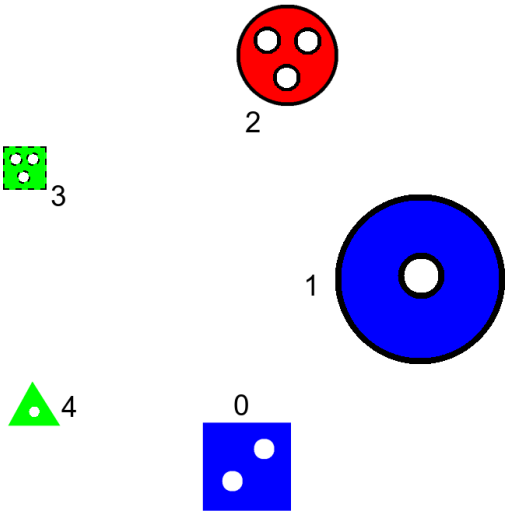
Solutions to most of the example ooops on the following pages can be found in section 4 on page 37.

3 Mathematical Formulation

This section is about treating Odd-one-out-puzzles as mathematical objects. The first part is about encoding the puzzle as a set of features, the features being partitions of the set of objects. This part requires only elementary naïve set theory. The second part is more technical and deals with the relationship of the different ways of defining what "solving" such a puzzle means. You can safely skip it as the only concept from it mentioned elsewhere in this paper without further explanation is the automorphism group.

3.1 Encoding as a Set of Partitions

For a given puzzle, let us call the collection of objects Σ, and the collection of features O. The rule that all objects stand on equal footing means that Σ is a set, i.e.



Label	size	color	shape	border	holes
0	small	blue	square	absent	2
1	big	blue	circle	black	1
2	medium	red	circle	black	3
3	small	green	square	dashed black	3
4	small	green	triangle	absent	1

Figure 2: A little harder. The numerical labels are for establishing correspondence with the textual description.

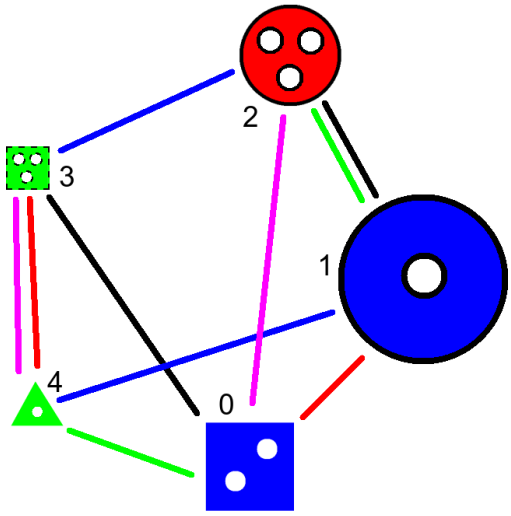


Figure 3: Still too easy. You don't even need to use different line colors for different features, but I did it anyway to make clear how the lines arise. Black lines stand for same shape, red is for color, green is for border style, blue is for hole count, and magenta is for size.

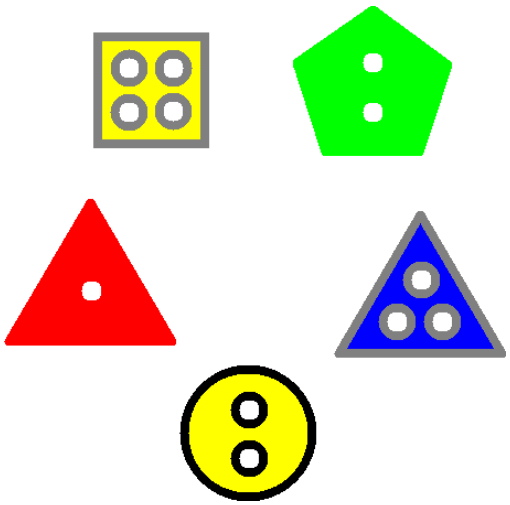
an unordered collection which contains each element at most once. We may denote its elements by consecutive natural numbers $0, 1, \ldots, n-1$. Note that integers are used here merely as convenient labels. It is invalid[1] to compare them to integers obtained in any way other than extracting them from Σ and O as set elements[2] by universal or existential quantification, and they may only be compared among each other for equality[3]. This is important because if we never break this symmetry, we can be sure that the algorithms and invariants developed later indeed depend on the abstract structure of the puzzle only and not on the concrete choice of integer labels.

The rules that all feature expressions stand on an equal footing and that only expressions of the same feature can be compared (and only for equality) means that it is natural to model a feature f as an equivalence relation $\simeq_f$ on Σ, or equivalently as a partition of Σ. A partition of a set Σ is simply a set of subsets (called *blocks*) of Σ that

[1]Or, to use the term introduced later, "not well-defined"

[2]Or, transitively, as elements of elements, elements of elements of elements, ...

[3]In set theoretic terms, they are urelements and there is no axiom of choice. In programming language terms, you can think of them as objects of an opaque datatype with a non-public constructor that are comparable for equality only, with no other functions provided and no choice operator on sets.



color	shape	border	holes
yellow	circle	black	2
blue	triangle	grey	3
green	pentagon	absent	2
yellow	square	grey	4
red	triangle	absent	1

Figure 4: One of the easiest I have found. It has only 4 features, and one of the features represents a partition of 5 different from the other features. There is only one object that has a unique expression of this feature, so that object is the solution.

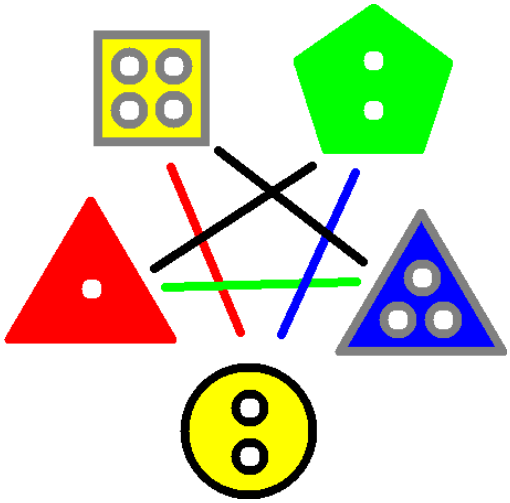


Figure 5: The graphical solution approach. This time, it was necessary to color the lines connecting equal feature expressions according to the respective feature to see a difference: Red is for color, green is for shape, blue is for hole count and black is for border.

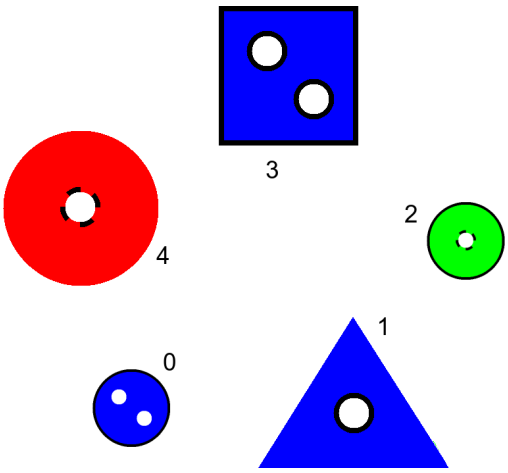
are disjoint and together contain all elements of Σ. Thus, we will model a feature to be a partition f whose blocks are the sets of objects that have the same expression of the feature modeled by f. When talking about a feature expression in isolation, it can then be modeled as a dependent pair $(f \in O, b \in f)$.

The rules that all features stand on an equal footing and that perfectly correlated features are actually just one and the same feature imply that O is a set.

Thus, the essence of an Odd-one-out puzzle with n objects that each have m features can be specified by O alone, where O is a set of m partitions of an n-element set Σ. For example, the puzzle from Figure 1 on page 28 could be written as

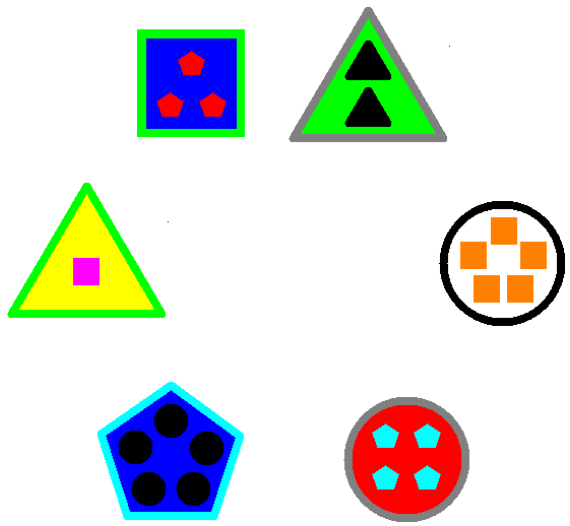
$$\left\{ \underbrace{\{\{0,1,3,4\},\{2\}\}}_{\text{size}}, \underbrace{\{\{0,1,2,4\},\{3\}\}}_{\text{color}}, \underbrace{\{\{0,1,2,3\},\{4\}\}}_{\text{shape}}, \underbrace{\{\{0,2,3,4\},\{1\}\}}_{\text{border}} \right\}$$

or shorter: (0123|4)(0124|3)(0134|2)(0234|1), with the solution evidently being 0 as it is the only object that does not occur alone in a block. We will use the shorter presentation



#	size	color	shape	border	holes	hole border
0	small	blue	circle	black	2	absent
1	big	blue	triangle	absent	1	black
2	small	green	circle	black	1	dashed
3	big	blue	square	black	2	black
4	big	red	circle	absent	1	dashed

Figure 6: Note that one object has the same style on its outer border and its hole borders. This doesn't make it special, because feature expressions may only be compared to expressions of the same feature.



color	shape	border	holes	hole color	hole shape
blue	pentagon	cyan	5	black	circle
red	circle	grey	4	cyan	pentagon
white	circle	black	5	orange	square
green	triangle	grey	2	black	triangle
blue	square	green	3	red	pentagon
yellow	triangle	green	1	magenta	square

Figure 7: An ooop with 6 objects and 6 features.

in the following, and the solution will not always be 0, in order to not spoil the riddle.

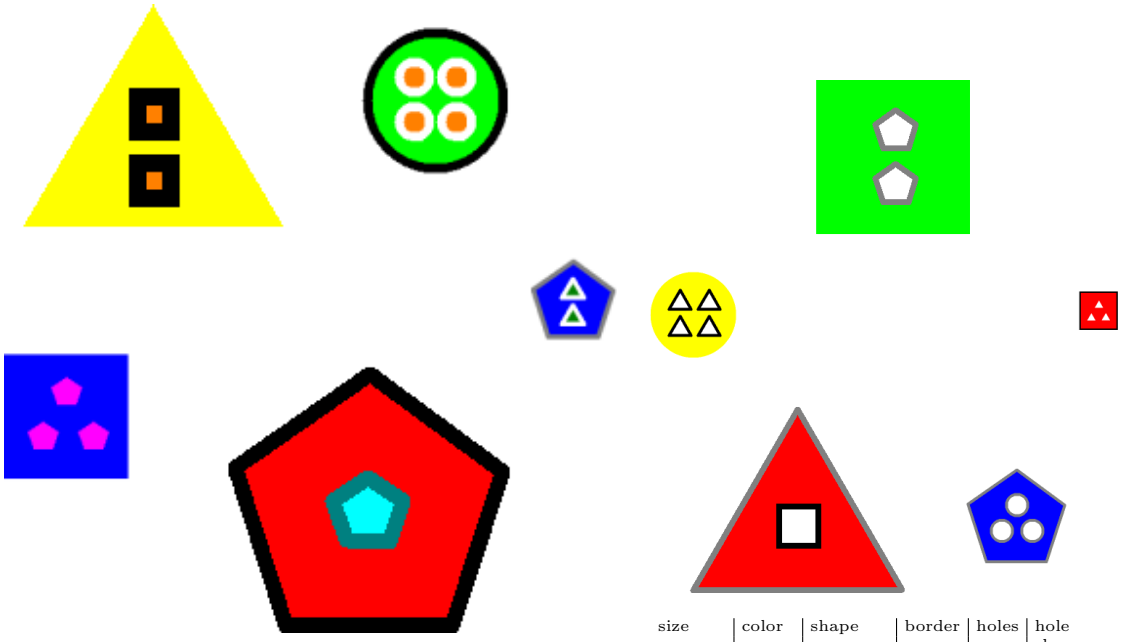
I define an *ooop*[4] O over Σ to be a set of partitions of Σ that is uniquely solvable. O being uniquely solvable means that there must be a uniquely special object $e \in \Sigma$ that is the solution. "Uniquely special" means that for each predicate P on Σ that is *well-defined in terms of* O (definable purely in terms of Σ, O, and the equality and set membership predicate, as well as the usual logical operators and quantifiers), it must hold that if P is true on exactly one object o, then $o = e$, and furthermore, the predicate $1_{\{e\}}$ that is true on e only must be well-defined.

Note that the name *ooop* is reserved for sets of partitions that are uniquely solvable. Other sets of partitions of Σ may be called "pre-ooops".

3.2 Ooop Solutions and Automorphisms

Feel free to skip this part, as it is quite technical and at the same time not totally rigorous.

[4]Odd-one-out puzzle



size	color	shape	border	holes	hole color	hole shape	hole border
big	red	pentagon	black	1	cyan	pentagon	dark cyan
small	blue	pentagon	gray	2	dark green	triangle	white
medium	green	circle	black	4	orange	circle	white
big	yellow	triangle	absent	2	orange	square	black
medium	blue	square	absent	3	magenta	pentagon	absent

Figure 8: Here, the automorphism group is D_4, the symmetry group of a square. It is a bit hard to see, but the interior of the triangles within the blue pentagon is dark green.

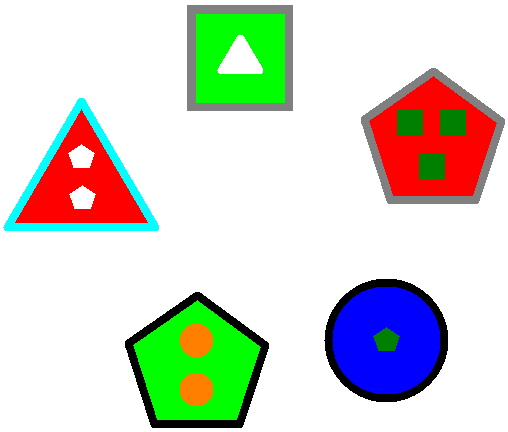
size	color	shape	border	holes	hole shape
big	red	triangle	grey	1	square
medium	blue	pentagon	grey	3	circle
small	red	square	black	3	triangle
big	green	square	absent	2	pentagon
medium	yellow	circle	absent	4	triangle

Figure 9: Here, the automorphism group is the symmetric group S_4 of all permutations of the non-special objects.

For O to be an Ooop, there must be exactly one predicate on Σ that is well defined in terms of O and true on exactly one object e. This is certainly the case if e exists as the unique common fixed point of all automorphisms of the puzzle, in a sense that will be explained now. What are the automorphisms? The only arbitrary choice in our presentation of the ooop as the set O was the assignment of the n integer labels to the objects. So each automorphism[5] of O is given by a permutation $\pi : \Sigma \to \Sigma$. Such a permutation π is an automorphism of O iff $\pi(O) = O$. Here I am using the convention that a function $f : A \to B$ is also a function of type $2^A \to 2^B$, that is, between the power sets of A and B[6], via $\forall\, S \subseteq A : f(S) = \{f(x) \mid x \in S\}$. One easily checks that these permutations indeed form a group.

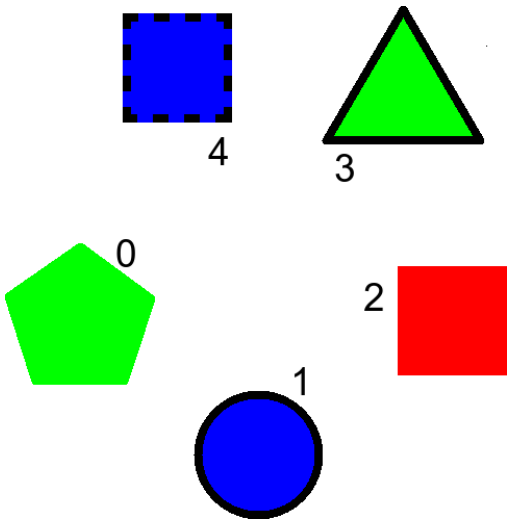
[5]This definition of automorphism applies not only to ooops, but also to other sets of partitions of Σ.

[6]Apply this convention thrice to see how to apply a permutation of Σ to a set of partitions of Σ (which is an element of $2^{2^{2^\Sigma}}$).



color	shape	border	holes	hole color	hole shape
green	pentagon	black	2	orange	circle
blue	circle	black	1	dark green	pentagon
red	pentagon	grey	3	dark green	square
green	square	grey	1	white	triangle
red	triangle	cyan	2	white	pentagon

Figure 10: Here, the automorphism group is C_4, the cyclic group of order 4.



#	color	shape	border
0	green	pentagon	absent
1	blue	circle	black
2	red	square	absent
3	green	triangle	black
4	blue	square	dashed

Figure 11: The algorithm from subsection 5.4 on page 40 takes multiple steps on this one.

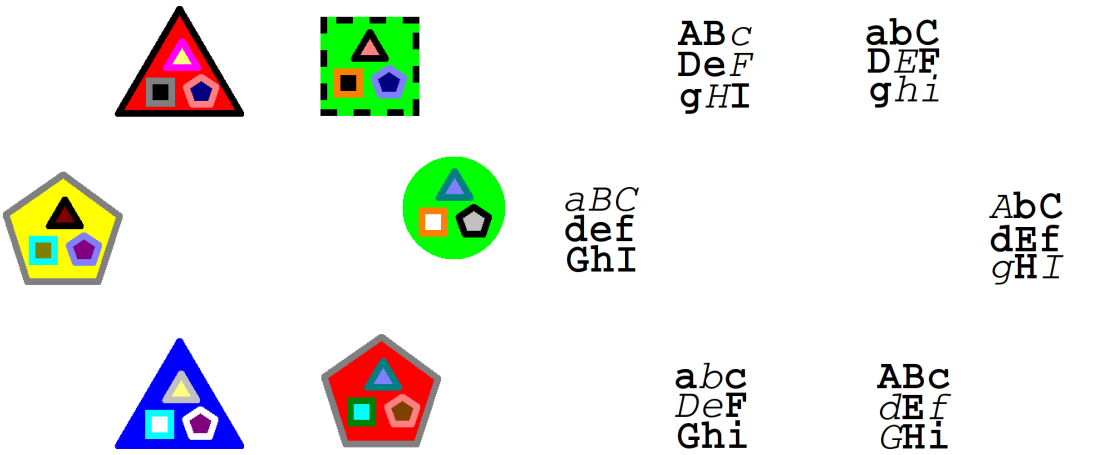


Figure 12: The hardest (yet). Cannot be solved by the incomplete algorithm (without using the characteristic polynomial as an invariant). By being a bit creative, I could manually find the hidden asymmetry without resorting to the automorphism group or the costly characteristic polynomial. It turns out that three of the features are special in the way they relate to other features. The ooop is also presented here with letter/typographic emphasis features for clearer representation.

Automorphisms have the property that they preserve well-defined predicates, or expressed differently, the notion of well-definedness in terms of O is invariant under automorphisms (because O is invariant). But what does a general permutation of Σ do to a well-defined predicate and to the notion of well-definedness? Let P_O be a predicate that is well-defined in terms of an ooop O, and let $P_{\pi(O)}$ be the predicate that is defined in the same way, but using $\pi(O)$ instead of O, for some permutation $\pi : \Sigma \to \Sigma$. If $O \neq \pi(O)$, then it is possible that $\exists\, i : P_O(i) \neq P_{\pi(O)}(i)$. It is however always the case that $\forall\, i : P_O(i) = P_{\pi(O)}(\pi(i))$ for all permutations π; this is because the notion of well-definedness is parameterized by the ooop and has no other way to treat two objects differently than by exploiting their structural role in the ooop, so if the object labels change by the same permutation in the presentation of the ooop as they do in the argument of the predicate, a well-defined predicate parameterized by the ooop must yield the same result. Hence, we can say that general permutations do not leave all well-defined predicates unchanged; well-defined predicates should be called *covariant* under permutations, but are in general not *invariant*. But if π is an automorphism of O, we have $O = \pi(O)$ and the two predicates are obviously the same: $\forall\, i : P_O(i) = P_{\pi(O)}(i)$, and so the notion of well-definedness, being parameterized by O, does not change under the automorphism and the predicate is preserved.

At the same time, the predicate is also covariant, and this leads to the following argument that a special object must be a common fixed point of all automorphisms: if there is an automorphism h of O that maps an object i to a different object j, none of the two objects can possibly be the special object e because $h(O) = O$, but $h(i) = j$, so i would fulfill the same predicates on Σ that are well-defined in terms of O as j does, because any Predicate P_O, well-defined in terms of O, is both covariant and invariant under automorphisms: $P_O(i) = P_{h(O)}(h(i)) = P_O(j)$. Hence there can be no well-defined predicate that has different truth values on i and j. But that would be required of the predicate $1_{\{i\}}$ that must be well-defined according to the definition of an ooop if i was the solution. The assumption that j is the solution leads to the same kind of contradiction. Hence, an object that is the solution of the ooop must necessarily be a fixed point of every automorphism. The predicate "Is a common fixed point of all automorphisms" is well-defined, and if it is true for exactly one object e, it is the required predicate $1_{\{e\}}$ and e is the solution.

Note that the preceding did neither claim nor prove that there cannot be ooops with multiple common fixed points of all automorphisms. I argue that this is the case: being a common fixed point of all automorphisms means being structurally different[7] from all other objects. Whenever there are two objects that are not structurally different, there is an automorphism that maps one to the other, and vice versa. For structurally different objects, however, it should always be possible to construct a well-defined predicate that tells them apart, because they play a different role in the structure of the set of partitions

[7]In the sense of playing a different role in the structure of the set of partitions

and the difference of their roles can be specified in a well-defined way by using that structure. So especially for the set of fixed points, there are then well-defined predicates that can tell each one apart from the rest.

For example, the pre-ooop $(0|12|34)(1|0234)$ has only one nontrivial automorphism, namely the transposition of 3 and 4. The three fixed points $i = 0, 1, 2$ are each characterized by a different predicate $1_{\{i\}}$ that is true exclusively on i:

- $1_{\{0\}}(j)$ can be expressed as "j is fixed by all automorphisms and does not occur in a block of size 2".
- $1_{\{1\}}(j)$ can be expressed as "j is fixed by all automorphisms and does not occur in a block of size 4".
- $1_{\{2\}}(j)$ can be expressed as "j is fixed by all automorphisms and does not occur in a block of size 1".

But according to the definition of an ooop, there can be only one such predicate, so that pre-ooop is not an ooop.

I omit a more rigorous proof that for each common fixed point i of all automorphisms, the predicate $1_{\{i\}}$ is well-defined.

4 About the Ooops in this Paper

Figure 2 on page 31 shows the only puzzle I have constructed by hand. The solution is given by the blue square, and one possible justification for this is "It is the only object so that for each other object, they agree on the expression of exactly one feature." Figure 3 on page 31 shows a graphical way to arrive at this statement.

Figure 4 on page 32 shows an ooop found by a computer search. The solution is given by the yellow circle, and one possible justification for this is "It is the only object that has a unique expression of the only feature (namely, "border style") that has three different expressions (instead of four)." Figure 5 on page 32 shows a graphical way to arrive at this solution.

Figures 6 and 7 on page 33 show two more puzzles. Solutions are intentionally not given.

For three of the following four ooops, I give only a text form of the justification of the solution. It should be easy to use the given justification to identify the special object. It should also be easy to ignore the texts in case you want to solve the riddles yourself.

A simple justification for the solution of Figure 8 on page 34 is: "There are two features that have only three expressions. The special object is the only one that has a unique expression of these two features."

A simple justification for the solution of Figure 9 on page 34 is: "There are three features that have only three expressions. The special object is the only one that has a unique expression of these three features."

The special object in Figure 10 on page 35 is the green square with grey border and one white triangular hole. What is the justification for this?

A simple justification for the solution of Figure 11 on page 35 is: "There is one feature that has four expressions. The special object is the only one that has nothing in common with any object that does not have a unique expression of this feature". The words "nothing in common" of course just apply to proper features; for example, all objects have their size and hole count in common, but these are not considered features here because they are the same for all objects in this ooop.

Finally, Figure 12 on page 35 shows the hardest ooop could find, with nine features. Note that due to the high number of features, I used two different sets of features and feature expressions. In the case of the Letter/Typographic emphasis features, each feature is assigned a letter and the four expressions of each feature are represented by four different ways of writing that letter. The graphical solution technique of drawing colored lines was in itself not sufficient to see which object is special.

5 Useful Definitions for Solving Ooops

For solving ooops, it is helpful to define some properties of objects or features that are invariant under automorphisms[8]. That is, if there is an automorphism that maps i to j, then i and j have the same value of the invariant property. This is also interesting for generating challenging ooops, because the easy solutions that could be solved using these invariants can quickly be weeded out or not even be considered in the first place.

5.1 Invariant: Class Histogram

The ooop depicted in Figure 1 on page 28 could be solved by counting how many feature expressions are unique to a given object i, and noting that this number is different for one object. This idea readily generalizes to a histogram of the sizes of the equivalence classes that i is a part of. This invariant I call the *class histogram*. To be precise, the class histogram for object i of an ooop O with n objects is a finite list of integers $C(O,i) \in \mathbb{N}^n$ with

$$C(O,i)_k = \left|\left\{ f \;\middle|\; i \in x \in f \in O, |x| = k \right\}\right|, \quad 1 \leqslant k \leqslant n$$

As an example, the class histogram for the solution of 1 on page 28 is $(0,0,0,4,0)$, while the class histograms for the other four objects are all $(1,0,0,3,0)$.

An ooop where all objects have the same class histogram is called *class constant*. To solve these, we consider other invariants.

[8] In case you did not read the previous subsection: an ooop-automorphism is essentially a relabeling of the objects (here: numbers) occurring in the set of partitions that leaves the representation of the riddle as a set of partitions unchanged.

5.2 Invariant: Link Histogram

The ooop depicted in Figure 2 on page 31 is class constant. It has presentation

$$\underbrace{(02|34|1)}_{\text{size}}\underbrace{(01|34|2)}_{\text{color}}\underbrace{(03|12|4)}_{\text{shape}}\underbrace{(04|12|3)}_{\text{border}}\underbrace{(0|14|23)}_{\text{holes}}$$

where the numbers are counting counterclockwise from the blue square. The class histogram for all objects is $(1, 4, 0, 0, 0)$.

To identify the special object, we need another invariant. If we draw a line between two objects for each feature for which the objects have the same expression, we arrive at Figure 3 on page 31, where the solution is obvious: most objects have two single lines and one double line attached to them, but one has four single lines. This motivates what I call the *link histogram*: it is a histogram that tells for each relevant number m how many partners an object has with which it agrees on m features.

The precise definition of the link histogram is that the link histogram of an object i of an ooop O is a finite list of integers $L(O, i) \in \mathbb{N}^{1+|O|}$ with

$$L(O,i)_k = \left| \left\{ j \;\middle|\; j \in \Sigma \setminus \{i\}, \left| \left\{ f \;\middle|\; f \in O, i \simeq_f j \right\} \right| = k \right\} \right|, \quad 0 \leqslant k \leqslant |O|$$

The link histograms for Figures 2 and 3 are: $(0, 4, 0, 0, 0, 0)$ for object 0 and $(1, 2, 1, 0, 0, 0)$ for all the rest. Thus we see object 0 can be singled out using the link histogram, and therefore is the solution. The natural language justification for the solution can be extracted from the link histogram: by observing that the only nonzero entry in the link histogram of the special object is at position 1, which means that it has 1 feature expression in common with each other object, we arrive at the description of the solution as "the only object so that for each other object, they agree on the expression of exactly one feature." (See section 4 on page 37)

An ooop where all objects have the same link histogram is called *link constant*.

5.3 Invariant: Integer Partitions

The ooop depicted in Figure 6 on page 33 is class constant and link constant. It has presentation[9]

$$\underbrace{(02|134)}_{\text{size}}\underbrace{(013|2|4)}_{\text{color}}\underbrace{(024|1|3)}_{\text{shape}}\underbrace{(023|14)}_{\text{outer border}}\underbrace{(03|124)}_{\text{holes}}\underbrace{(0|13|24)}_{\text{hole border}}$$

[9]Note that there are different ways to label the objects with numbers that result in the same presentation, but then the partitions of Σ would denote different features than they do now. This is an effect of the existence of automorphisms. For the most part, I avoid making such arbitrary choices because I want to encourage thinking about the objects as what they are in relation to each other and not as some kind of label that introduces a spurious distinctness.

The class histogram for all objects is $(1, 2, 3, 0, 0)$ and the link histogram is $(0, 2, 0, 2, 0, 0, 0)$. Therefore, it cannot be solved simply by using these invariants.

However, the features, viewed as partitions, do not represent the same integer partitions: The first two correspond to the integer partition $1+1+3$, the next three correspond to the integer partition $2 + 3$, and the last one corresponds to $1 + 2 + 2$. This census of integer partitions is obviously invariant under ooop-automorphisms (indeed, under arbitrary permutations of Σ). Retaining one of the three kinds of feature and discarding the rest does not break any symmetries that were not already broken and thus yields another ooop with the same solution, which may then be solvable recursively with this or another technique.

In the case of the ooop from Figure 6 on page 33, we can for example consider only the feature "hole border" $(14|0|23)$, as it is the only one corresponding to the integer partition $1 + 2 + 2$. In the resulting ooop $O' = \big\{\{0\}, \{1, 3\}, \{2, 4\}\big\}$, only one object e has $C(O', e)_1 = 1$, thus it is the solution. Of course this could be said more plainly by stating that the feature "hole border" is special because it is the only one that has a unique expression on exactly one object, thus that object, 0, must be the solution. My reason for first giving the more long-winded statement about using the integer partition invariant to justify ignoring some features, and then using the class histogram on the rest to single out an object, is that it generalizes more systematically to the algorithm explained in the next subsection.

An ooop where all features represent the same integer partition is called *partition constant*. Examples of partition constant ooops are shown in Figure 2 on page 31 (not listed in Table 2 on page 49, as it is not link constant) and Figure 12 on page 35 (presented in the last row of Table 2).

5.4 An Incomplete Solution Algorithm

A solution can always be found as the fixed point of the automorphism group's action on Σ, which can be found by checking all $n!$ permutations of Σ. But often a more direct way is available that does not suffer that much from the effects of combinatorial explosion, running in polynomial time if it terminates at all.

The idea here is to compute a sequence F_k of partitions of O and a sequence B_k of partitions of Σ. These partitions tell us what we already know about some features and objects being structurally different from others after iteration k of the algorithm. Initially, the partitions B_0 and F_0 contain only one block, because a priori all objects and also all features stand on an equal footing. Then, asymmetries are discovered using invariants and in turn induce discovery of other asymmetries until an asymmetry is found that singles out one object.

If you are not interested in the technical details of the algorithm and an example how it runs, you can skip the rest of the subsection.

We will need an operation $\vec{\cap}$ that, given a set M (such as an ooop or feature) and

a subset b of Σ, returns a version of M where only objects in b have been retained at whatever depth they occurred, and empty sets are removed. That is,

$$M \vec{\cap} b = (M \cap b) \cup \left\{ S \vec{\cap} b \;\middle|\; S \in M, S \notin \Sigma \right\} \setminus \{\emptyset\}$$

At iteration $k \in \mathbb{N}_+$ of the algorithm, we do the following:

- Refine the partition of O: F_k is computed from F_{k-1} and B_{k-1} as the coarsest common refinement of
  - F_{k-1}
  - For each $b \in B_{k-1}$, the partition induced by the equivalence relation $\sim_b$ on O defined as follows: $f \sim_b g$ iff the two partitions $f \vec{\cap} b$ resp. $g \vec{\cap} b$ of b represent the same partition of the integer $|b|$.
- Refine the partition of Σ: B_k is computed from F_k and B_{k-1} as the coarsest common refinement of
  - B_{k-1}
  - For each $(b, c) \in B_{k-1} \times F_k$, the union of $\Sigma \setminus b$ and the partition induced by the equivalence relation $\approx_{b,c}$ on b defined by

    $$i \approx_{b,c} j \iff C(c', i) = C(c', j) \wedge L(c', i) = L(c', j)$$

    where $c' = c \vec{\cap} b$.[10] Here, the definition of link histograms and class histograms has been extended to be applicable to sets of partitions of Σ that are not necessarily ooops.
- Test if B_k contains a block b of size 1. If so, output the element of b as the solution and terminate. Else, continue with iteration $k + 1$.

One can easily see that the algorithm is correct by observing that everything is covariant under permutation of the integers in the input ooop because they are only ever compared for equality. Therefore, if the algorithm can single out an element of Σ, it must be the solution, which is presumed to be unique. However, the algorithm cannot solve every ooop. The one shown in Figure 12 on page 35 and presented in the last row of Table 2 on page 50 is a counterexample because it is class constant, link constant and partition constant. Hence, the algorithm never refines anything.

[10] This distinguishes the objects by several class and link histograms that each consider one subset of features that we could already tell apart from others (but not from each other), and one subset of objects that we could already tell apart from others (but not from each other).

As an example run of the algorithm, let us solve the ooop displayed in Figure 11 on page 35. It has presentation

$$\underbrace{(2|03|14)}_{\text{color}}\underbrace{(02|13|4)}_{\text{border}}\underbrace{(3|0|1|24)}_{\text{shape}}$$

as seen in row 2 of Table 2 on page 49. The algorithm then runs as follows: (feel free to skip this unless you want to trace in detail how the algorithm works)

$$\begin{aligned}
F_0 &= \Big\{\Big\{\underbrace{\{\{0,3\},\{2\},\{1,4\}\}}_{f_{\text{color}}},\underbrace{\{\{0,2\},\{1,3\},\{4\}\}}_{f_{\text{border}}},\underbrace{\{\{0\},\{1\},\{2,4\},\{3\}\}}_{f_{\text{shape}}}\Big\}\Big\}\\
B_0 &= \big\{\{0,1,2,3,4\}\big\}\\
b_{1,1} &\coloneqq \{0,1,2,3,4\}\\
{\sim_{b_{1,1}}} &\equiv \Big\{\big\{\{\{0,3\},\{2\},\{1,4\}\},\{\{0,2\},\{1,3\},\{4\}\}\big\},\quad \big\{\{\{0\},\{1\},\{2,4\},\{3\}\}\big\}\Big\}\\
&= \Big\{\{f_{\text{color}},f_{\text{border}}\},\big\{f_{\text{shape}}\big\}\Big\}\\
F_1 &= \Big\{\big\{\{\{0,3\},\{2\},\{1,4\}\},\{\{0,2\},\{1,3\},\{4\}\}\big\},\quad \big\{\{\{0\},\{1\},\{2,4\},\{3\}\}\big\}\Big\}\\
&= \Big\{\{f_{\text{color}},f_{\text{border}}\},\big\{f_{\text{shape}}\big\}\Big\}
\end{aligned}$$

The feature $f_{\text{shape}} = \big\{\{0\},\{1\},\{2,4\},\{3\}\big\}$ has been determined to be structurally different from the other two.

$$\begin{aligned}
c_{1,1} &\coloneqq \big\{\{\{0,3\},\{2\},\{1,4\}\},\{\{0,2\},\{1,3\},\{4\}\}\big\} = \{f_{\text{color}},f_{\text{border}}\}\\
c'_{1,1} &\coloneqq c_{1,1} \vec{\cap} b_{1,1} = c_{1,1}\\
C(c'_{1,1},0) &= C(c'_{1,1},1) = C(c'_{1,1},3) = (0,2,0,0,0)\\
C(c'_{1,1},2) &= C(c'_{1,1},4) = (1,1,0,0,0)\\
L(c'_{1,1},0) &= L(c'_{1,1},1) = L(c'_{1,1},3) = (2,2,0)\\
L(c'_{1,1},2) &= L(c'_{1,1},4) = (3,1,0)\\
{\approx_{b_{1,1},c_{1,1}}} &\equiv \big\{\{0,1,3\},\{2,4\}\big\}
\end{aligned}$$

$$\begin{aligned}
c_{1,2} &\coloneqq \big\{\{\{0\},\{1\},\{2,4\},\{3\}\}\big\} = \big\{f_{\text{shape}}\big\}\\
c'_{1,2} &\coloneqq c_{1,2} \vec{\cap} b_{1,1} = c_{1,2}\\
C(c'_{1,2},0) &= C(c'_{1,2},1) = C(c'_{1,2},3) = (1,0,0,0,0)
\end{aligned}$$

$$C(c'_{1,2}, 2) = C(c'_{1,2}, 4) = (0, 1, 0, 0, 0)$$
$$L(c'_{1,2}, 0) = L(c'_{1,2}, 1) = L(c'_{1,2}, 3) = (4, 0, 0, 0, 0)$$
$$L(c'_{1,2}, 2) = L(c'_{1,2}, 4) = (3, 1, 0, 0, 0)$$
$$\approx_{b_{1,1}, c_{1,2}} \equiv \left\{\{0, 1, 3\}, \{2, 4\}\right\}$$

The objects $0, 1, 3$ have been determined to be structurally different from the objects $2, 4$.

$$B_1 = \left\{\{0, 1, 3\}, \{2, 4\}\right\}$$

Next iteration.

$$b_{2,1} \coloneqq \{0, 1, 3\}$$

$$\begin{array}{rrll} & f_{\text{color}} \vec{\cap} b_{2,1} = & \left\{\{0, 3\}, \{2\}, \{1, 4\}\right\} \vec{\cap} b_{2,1} & = \left\{\{0, 3\}, \{1\}\right\} \\ \text{Note:} & f_{\text{border}} \vec{\cap} b_{2,1} = & \left\{\{0, 2\}, \{1, 3\}, \{4\}\right\} \vec{\cap} b_{2,1} & = \left\{\{0\}, \{1, 3\}\right\} \\ & f_{\text{shape}} \vec{\cap} b_{2,1} = & \left\{\{0\}, \{1\}, \{2, 4\}, \{3\}\right\} \vec{\cap} b_{2,1} & = \left\{\{0\}, \{1\}, \{3\}\right\} \end{array}$$

$$\sim_{b_{2,1}} = \sim_{b_{1,1}}$$
$$b_{2,2} \coloneqq \{2, 4\}$$

$$\begin{array}{rrll} & f_{\text{color}} \vec{\cap} b_{2,2} = & \left\{\{0, 3\}, \{2\}, \{1, 4\}\right\} \vec{\cap} b_{2,2} & = \left\{\{2\}, \{4\}\right\} \\ \text{Note:} & f_{\text{border}} \vec{\cap} b_{2,2} = & \left\{\{0, 2\}, \{1, 3\}, \{4\}\right\} \vec{\cap} b_{2,2} & = \left\{\{2\}, \{4\}\right\} \\ & f_{\text{shape}} \vec{\cap} b_{2,2} = & \left\{\{0\}, \{1\}, \{2, 4\}, \{3\}\right\} \vec{\cap} b_{2,2} & = \left\{\{2, 4\}\right\} \end{array}$$

$$\sim_{b_{2,1}} = \sim_{b_{1,1}}$$
$$F_2 = F_1$$

No new information about the structural differences of features could be derived.

$$c_{2,1} \coloneqq c_{1,1}$$
$$c'_{2,1} \coloneqq c_{2,1} \vec{\cap} b_{2,1} = \left\{\left\{\{0, 3\}, \{1\}\right\}, \left\{\{0\}, \{1, 3\}\right\}\right\}$$
$$C(c'_{2,1}, 0) = C(c'_{2,1}, 1) = (1, 1, 0)$$
$$C(c'_{2,1}, 3) = (0, 2, 0)$$
$$L(c'_{2,1}, 0) = L(c'_{2,1}, 1) = (1, 1, 0)$$
$$L(c'_{2,1}, 3) = (0, 2, 0)$$
$$\approx_{b_{2,1}, c_{2,1}} \equiv \left\{\{0, 1\}, \{3\}, \{2, 4\}\right\}$$

At this point, an optimized implementation of the algorithm might already notice that the solution has been found. But let's continue, for completeness of presentation:

$$c'_{2,2} \coloneqq c_{2,1} \vec{\cap} b_{2,2} = \left\{\left\{\{2\}, \{4\}\right\}\right\}$$

$$
\begin{aligned}
C(c'_{2,2},2) &= C(c'_{2,2},4) = (1,0)\\
L(c'_{2,2},2) &= L(c'_{2,2},4) = (1,0)\\
\approx_{b_{2,2},c_{2,1}} &\equiv \left\{\{0,1,3\},\{2,4\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
c_{2,2} &\coloneqq c_{1,2}\\
c'_{2,3} &\coloneqq c_{2,2} \vec{\cap} b_{2,1} = \left\{\left\{\{0\},\{1\},\{3\}\right\}\right\}\\
C(c'_{2,3},0) &= C(c'_{2,3},1) = C(c'_{2,3},3) = (1,0,0)\\
L(c'_{2,3},0) &= L(c'_{2,3},1) = L(c'_{2,3},3) = (1,0)\\
\approx_{b_{2,1},c_{2,2}} &\equiv \left\{\{0,1,3\},\{2,4\}\right\}
\end{aligned}
$$

$$
\begin{aligned}
c'_{2,4} &\coloneqq c_{2,2} \vec{\cap} b_{2,2} = \left\{\left\{\{2,4\}\right\}\right\}\\
C(c'_{2,4},2) &= C(c'_{2,4},4) = (0,1)\\
L(c'_{2,4},2) &= L(c'_{2,4},4) = (0,1)\\
\approx_{b_{2,2},c_{2,2}} &\equiv \left\{\{0,1,3\},\{2,4\}\right\}\\
B_2 &= \left\{\{0,1\},\{3\},\{2,4\}\right\}
\end{aligned}
$$

The algorithm terminates and has found the solution 3, because 3 is in a block all for itself.

5.5 Invariant: Characteristic Polynomial

This subsection is again quite technical and requires advanced knowledge of linear algebra and some graph theory and number theory. It is not relevant to the rest of the paper and can safely be skipped.

The invariants discussed so far are all based on local properties of an object or feature: they do not look further than the immediate surroundings in the network of relationships that these entities have among each other. A global approach should yield stronger invariants.

Here is one idea to construct such an invariant, applicable to both objects o and features p: use the characteristic polynomial of a multigraph whose vertices are the feature expressions $(f \in O, x \in f)$ and which has an edge between two feature expressions (f,x) and (g,y) whenever $f = g \ \wedge \ x \neq y$ and also for each $z \in (x \cap y)$ whenever $f \neq g$ (note that the latter is meant to result in $|x \cap y|$-fold edges). Additionally, a feature

expression (f, x) gets a self-loop iff $f = p$ (in case the invariant is computed for a feature p), or iff $o \in x$ (in case the invariant is computed for an object o). Note that treating an object or feature different from the others (via the self-loops) does not violate the requirement that symmetry not be broken, provided we do it for all separately in the same way for all objects or features, and combine the results in a symmetric manner. These three edge types correspond to three relevant relations that feature expressions can have: The binary relations "expressions of the same feature" and "expressed on the same object", and the unary relation "is an expression of the feature that the characteristic polynomial is computed for" resp. "is expressed on the object that the characteristic polynomial is computed for". To obtain a stronger invariant, the three edge types are represented by different values in the adjacency matrix of the graph.

The adjacency matrix for a graph that treats one object o differently has rows and columns indexed by feature expressions $(f \in O, x \in f)$ and its entries are polynomials in a, b, c:

$$M^{(o)}_{(f,x)(g,y)} = \begin{cases} a & f = g \wedge x = y \wedge o \in x \\ b & f = g \wedge x \neq y \\ |x \cap y| \cdot c & f \neq g \\ 0 & \text{otherwise} \end{cases}$$

The adjacency matrix $M^{(p)}$ for a graph that treats one feature p differently, on the other hand, has entries:

$$M^{(p)}_{(f,x)(g,y)} = \begin{cases} a & f = g \wedge x = y \wedge f = p \\ b & f = g \wedge x \neq y \\ |x \cap y| \cdot c & f \neq g \\ 0 & \text{otherwise} \end{cases}$$

Thus, the characteristic polynomial $\det\left(M^{(\cdot)} - \lambda \cdot \mathbb{1}\right)$ of an object or feature is a polynomial over the integers in the four variables a, b, c and λ. It may be used in an attempt to tell the objects apart directly or in the context of the algorithm from the previous section as an additional invariant to tell apart objects (along with or instead of the invariants C and L) or features (along with or instead of the represented integer partition).

The full characteristic polynomial is costly to compute and usually not necessary to find out special objects or features. The following simplifications can be made to get an invariant that is still useful:

- Substitute constants for a, b, c to get an univariate polynomial.

- Compute in a finite field F_q for some prime q.
- Compute only the residue of the polynomials modulo some irreducible polynomial.

In the following example, the first two simplifications have been applied, with

$$a = 11, b = 13, c = 17, \text{ and } q = 2^{31} - 1$$

to compute the characteristic polynomials for the ooop shown in Figure 12 on page 35 and presented in the last row of Table 2 on page 50. This ooop is the only known ooop[11] that is class constant, link constant and partition constant, and therefore could not be solved by the incomplete algorithm from the previous subsection.

For one object, the characteristic polynomial obtained this way is

$$1000686794+778979638\lambda+1208471834\lambda^2+214842245\lambda^3+407159371\lambda^4+2061933376\lambda^5+$$
$$+220672646\lambda^6+1901760362\lambda^7+2080891101\lambda^8+1546271272\lambda^9+1253466070\lambda^{10}+$$
$$+1331302039\lambda^{11}+1189184501\lambda^{12}+391674026\lambda^{13}+1229795532\lambda^{14}+176746726\lambda^{15}+$$
$$+811071339\lambda^{16}+1665817037\lambda^{17}+412811464\lambda^{18}+1403554725\lambda^{19}+978000731\lambda^{20}+$$
$$+1184024975\lambda^{21}+1238899225\lambda^{22}+1699918815\lambda^{23}+1895673066\lambda^{24}+772980284\lambda^{25}+$$
$$+188615753\lambda^{26}+784633410\lambda^{27}+1856783405\lambda^{28}+1954202388\lambda^{29}+1052075028\lambda^{30}+$$
$$+2002069589\lambda^{31}+1892952859\lambda^{32}+580162601\lambda^{33}+1902443869\lambda^{34}+2147483548\lambda^{35}+1\lambda^{36}$$

For two of the objects, the characteristic polynomial is

$$199804076+946132397\lambda+2019487070\lambda^2+1155996283\lambda^3+117794738\lambda^4+693807738\lambda^5+$$
$$+1885542751\lambda^6+756373839\lambda^7+1701861809\lambda^8+795190249\lambda^9+154076335\lambda^{10}+$$
$$+1498806526\lambda^{11}+1565519423\lambda^{12}+1327407924\lambda^{13}+343566432\lambda^{14}+1889794430\lambda^{15}+$$
$$+617814599\lambda^{16}+1797185656\lambda^{17}+833959024\lambda^{18}+658492377\lambda^{19}+1856809957\lambda^{20}+$$
$$+639642216\lambda^{21}+133381603\lambda^{22}+1564465521\lambda^{23}+587228759\lambda^{24}+1731230403\lambda^{25}+$$
$$+2126664317\lambda^{26}+1183731482\lambda^{27}+1691807419\lambda^{28}+984612598\lambda^{29}+642095191\lambda^{30}+$$
$$+2002069589\lambda^{31}+1892952859\lambda^{32}+580162601\lambda^{33}+1902443869\lambda^{34}+2147483548\lambda^{35}+1\lambda^{36}$$

For the remaining three objects, the characteristic polynomial is

$$1781456468+1259785295\lambda+971138483\lambda^2+1183570439\lambda^3+920726353\lambda^4+2141349017\lambda^5+$$

[11] Apart from the trivial $\Big\{\{\{0\}\}\Big\}$

$$+229538027\lambda^6+1501555664\lambda^7+1765004535\lambda^8+394725299\lambda^9+1200544688\lambda^{10}+$$
$$+1557373746\lambda^{11}+2076428178\lambda^{12}+681627508\lambda^{13}+1535636483\lambda^{14}+1508409635\lambda^{15}+$$
$$+869115566\lambda^{16}+1357296231\lambda^{17}+771828013\lambda^{18}+894415995\lambda^{19}+1049757853\lambda^{20}+$$
$$+888659133\lambda^{21}+562287643\lambda^{22}+448010323\lambda^{23}+829518313\lambda^{24}+128655495\lambda^{25}+$$
$$+460784954\lambda^{26}+590599328\lambda^{27}+2001018801\lambda^{28}+15022808\lambda^{29}+232115354\lambda^{30}+$$
$$+2002069589\lambda^{31}+1892952859\lambda^{32}+580162601\lambda^{33}+1902443869\lambda^{34}+2147483548\lambda^{35}+1\lambda^{36}$$

So we see that in this case, using the simplified characteristic polynomial directly as an invariant for objects was sufficient for finding the odd one out. Actually, the constant terms of the polynomials would already have been enough, and these are just the determinants of the matrices, so in this case there was not even a need to handle any polynomials. This demonstrates that the determinants of the adjacency matrices are already on their own useful invariants that detect differences not detected by the class and link histograms.

For completeness, here are the characteristic polynomials for the features:

Three features have characteristic polynomial

$$1103825254+271709124\lambda+940632656\lambda^2+1321200813\lambda^3+275847580\lambda^4+639479212\lambda^5+$$
$$+2006258607\lambda^6+435937673\lambda^7+250669347\lambda^8+609865962\lambda^9+463063716\lambda^{10}+$$
$$+1149682437\lambda^{11}+1546682903\lambda^{12}+98121445\lambda^{13}+78408214\lambda^{14}+1361178359\lambda^{15}+$$
$$+376397978\lambda^{16}+1000588474\lambda^{17}+1845784957\lambda^{18}+1112872939\lambda^{19}+1267952900\lambda^{20}+$$
$$+1078281656\lambda^{21}+1313932574\lambda^{22}+970420845\lambda^{23}+605822464\lambda^{24}+182143317\lambda^{25}+$$
$$+683412853\lambda^{26}+1840622570\lambda^{27}+658162437\lambda^{28}+138670906\lambda^{29}+373019954\lambda^{30}+$$
$$+1899964665\lambda^{31}+1605503986\lambda^{32}+873211990\lambda^{33}+1902440239\lambda^{34}+2147483603\lambda^{35}+1\lambda^{36}$$

The remaining six have characteristic polynomial

$$1470001302+89131162\lambda+839884428\lambda^2+1349959666\lambda^3+2064892978\lambda^4+1025553272\lambda^5+$$
$$+48155734\lambda^6+1118871747\lambda^7+105292165\lambda^8+657832426\lambda^9+548252311\lambda^{10}+$$
$$+1172672363\lambda^{11}+911929140\lambda^{12}+1336291527\lambda^{13}+1281178238\lambda^{14}+1711599975\lambda^{15}+$$
$$+1148307748\lambda^{16}+985000829\lambda^{17}+942289526\lambda^{18}+95601140\lambda^{19}+1668443904\lambda^{20}+$$
$$+464649528\lambda^{21}+1939404040\lambda^{22}+1421992093\lambda^{23}+1584353662\lambda^{24}+588942694\lambda^{25}+$$
$$+1088038517\lambda^{26}+1034399759\lambda^{27}+1350393884\lambda^{28}+1667618633\lambda^{29}+373019954\lambda^{30}+$$

$+1899964665\lambda^{31}+1605503986\lambda^{32}+873211990\lambda^{33}+1902440239\lambda^{34}+2147483603\lambda^{35}+1\lambda^{36}$

Here, too, the constant terms are already sufficient to note a difference. This suggests a different way to use the characteristic polynomial: instead of actually computing the symbolic expression for the polynomial, which is quite a hassle as it involves the determinant of a polynomial-valued matrix, we simply evaluate the polynomial at different values of λ, one after the other. This requires just an ordinary determinant computation with simple values for each evaluation. If we are lucky, the first evaluation already gives us the solution or at least new information. Else, we evaluate at a different value of λ, for up to $m \coloneqq \sum_{f \in O} |f|$ different values of λ. Because m is the degree of the polynomial (which is monic), this gives us all the information and discriminative power of the polynomial, but incrementally. After that, more evaluations do not give us new information, as the polynomial can already be reconstructed via interpolation from the first m evaluations.

I suspect that the graph contains some redundancy[12] and the degree of the characteristic polynomial may be made lower without losing discriminative power. For example, by omitting from the graph the nodes for feature expressions that occur on only one object, i.e. nodes (f, x) with $|x| = 1$, no information might be lost. It might be possible to infer their existence and the edges in which they participate from the remaining nodes and their relations to each other, because for each feature f, being a partition of n elements, it must hold that $\sum_{x \in f} |x| = n$, and if we know that any missing nodes stand for unique feature expressions, we can maybe reconstruct the graph with these nodes from the one without them.

6 A Table of Ooops

Table 2 on the next page lists all class constant[13], link constant[14] ooops with n objects with at most $n + 3$ features for $n \leqslant 5$, found with brute force search by a computer. It turns out that there are none with $n < 5$. Features that have exactly 1 or n equivalence classes of expressions have been excluded from consideration, as they do not contain any information that could help with solving the puzzle. You can of course still use them in a graphical version of the puzzle to sow confusion.

Because the the ooops in the table are class constant and link constant, the class histograms[15] and link histograms[16] for all objects in a table row are the same. The two histograms are given in a separate column each. Since all numbers occurring in the

[12] Compare the higher order terms of the polynomials and observe that there is not much variation to see what raised my suspicions.

[13] See subsection 5.1 on page 38

[14] See subsection 5.2 on page 39

[15] See subsection 5.1 on page 38

[16] See subsection 5.2 on page 39

histograms have only one digit, the histograms (columns L and C) are simply given as a string of digits with trailing zeros removed to save space.

Of course, only one of each isomorphism class of ooops is listed. Two ooops O and O' over Σ are isomorphic iff there is a permutation $\pi : \Sigma \to \Sigma$ with $\pi(O) = O'$.

Some of the table rows have an associated graphical depiction in this paper. A correspondence of the numbers and partitions in the table to the objects and features in the images is usually not given as it would not be unique anyway due to the existence of nontrivial automorphisms.

Most of the automorphism groups are isomorphic to C_2, except for one occurrence of D_4 (depicted in Figure 8 on page 34), one occurrence of C_4 (Figure 10 on page 35) and one occurrence of S_4 (Figure 9 on page 34), which is the maximum possible automorphism group for an ooop with 5 objects, as there are no further ways to permute the 4 non-special elements while leaving the special element fixed.

Additionally, two ooops with $n = 6$ are listed. Both of them have automorphism group $C_2 \times S_3$. In one of them, all features correspond to the same integer partition $1 + 1 + 2 + 2$. This makes it partition constant[17] in addition to being class constant and link constant. Therefore, the solution techniques presented in section 5 on page 38 are insufficient to solve it, except for the characteristic polynomial. It is depicted in Figure 12 on page 35.

Table 2: List of all known class-constant and link-constant ooops

Presentation (object labels chosen randomly)	Symmetry	C	L	Figure
(4|02|13)(1|4|2|03)(1|0|24|3)(14|0|2|3)	C_2	22	22	4
(2|03|14)(02|13|4)(3|0|1|24)	C_2	12	22	11
(1|24|03)(0|24|13)(14|02|3)(1|4|2|03)(1|4|02|3)(4|0|2|13)(14|0|2|3)	C_2	24	202	
(01|2|34)(01|24|3)(0|12|34)(1|24|03)(1|4|2|03)(4|0|12|3)	C_2	24	202	
(1|02|34)(1|24|03)(14|0|2|3)(01|4|2|3)(1|04|2|3)(4|0|2|13)(4|0|12|3)(1|4|0|23)	D_4	44	04	8
(14|0|23)(01|24|3)(1|04|2|3)(1|0|2|34)(1|4|2|03)(4|0|2|13)(1|4|02|3)(4|0|12|3)	C_2	44	04	
(14|0|23)(0|12|34)(0|24|13)(01|4|2|3)(1|04|2|3)(1|4|02|3)(1|4|2|03)	S_4	34	04	9
(14|2|03)(0|12|34)(01|24|3)(1|4|02|3)(1|4|0|23)(4|0|2|13)(1|04|2|3)	C_2	34	04	
(14|02|3)(1|04|23)(4|12|03)(01|24|3)(4|0|2|13)(1|0|2|34)	C_2	24	04	
(0|24|13)(14|02|3)(1|04|23)(4|12|03)(1|0|2|34)(01|4|2|3)	C_4	24	04	10
(1|24|03)(4|02|13)(0|24|13)(01|2|34)(01|4|23)(04|12|3)(14|0|23)(1|04|2|3)	C_2	26	022	
(1|24|03)(14|0|23)(14|2|03)(04|12|3)(0|12|34)(1|04|23)(4|02|13)(01|4|2|3)	C_2	26	022	
(04|12|3)(01|2|34)(1|02|34)(1|24|03)(14|02|3)(14|0|23)(01|4|23)(4|0|2|13)	C_2	26	022	
(14|0|23)(4|12|03)(1|24|03)(1|02|34)(14|02|3)(0|24|13)(04|2|13)(01|4|2|3)	C_2	26	022	
(4|012|3)(0|2|134)(1|2|034)(1|024|3)(4|0|123)(4|02|13)(0|12|34)(1|04|2|3)	C_2	323	0202	
(1|4|023)(0|2|134)(1|2|034)(4|0|12|3)(14|0|2|3)(1|4|02|3)(012|34)(124|03)	C_2	323	0202	
(0|2|134)(4|012|3)(014|2|3)(1|02|34)(1|4|0|23)(01|234)(14|023)	C_2	223	0202	
(4|012|3)(1|0|234)(1|2|034)(014|2|3)(01|2|34)(1|4|0|23)(4|0|12|3)(04|123)	C_2	323	0202	
(4|2|013)(1|0|234)(0|124|3)(4|012|3)(01|2|34)(1|24|03)(12|034)	C_2	223	0202	
(0|124|3)(1|2|034)(1|4|2|03)(4|0|12|3)(014|23)(04|123)(14|023)	C_2	223	0202	

[17] See subsection 5.3 on page 40

Presentation	Symmetry	C	L	Figure
(02|134)(013|2|4)(024|1|3)(023|14)(03|124)(0|13|24)	C_2	123	0202	6
(1|024|3)(1|02|34)(4|2|013)(014|2|3)(4|12|03)(4|0|123)(1|0|234)(14|0|2|3)	C_2	323	004	
(1|2|034)(1|4|023)(0|124|3)(1|4|2|03)(14|0|2|3)(4|0|12|3)(04|123)(014|23)	C_2	323	0202	
(4|0|123)(1|04|2|3)(124|03)(24|013)(024|13)(12|034)	C_2	123	0202	
(4|12|03)(1|24|03)(0|124|3)(01|2|34)(01|234)(012|34)(12|034)(24|013)	C_2	143	02002	
(4|2|013)(0|2|134)(1|4|02|3)(1|0|24|3)(124|03)(14|023)(024|13)	C_2	233	0202	
(1|24|03)(04|2|13)(4|0|12|3)(024|13)(24|013)(12|034)(124|03)(04|123)	C_2	143	02002	
(14|0|23)(4|012|3)(1|24|03)(01|2|34)(04|123)(024|13)(02|134)(12|034)	C_2	143	0022	
(01|2|34)(0|24|13)(1|4|02|3)(12|034)(04|123)(14|023)(014|23)(124|03)	C_2	143	0022	
(04|12|3)(01|4|23)(1|0|2|34)(0|1234)(2|0134)(4|0123)(0124|3)(1|0234)	C_2	2204	00022	
(01|5|4|23)(4|05|2|13)(1|4|02|35)(14|5|0|2|3)(1|45|0|2|3)(1|5|0|24|3)	$C_2 \times S_3$	33	23	7
(1|4|05|23)(4|05|12|3)(4|0|12|35) (14|5|02|3)(15|04|2|3)(14|5|0|23) (1|5|02|34)(15|0|2|34)(1|04|2|35)	$C_2 \times S_3$	36	203	12

7 Philosophical Remarks

The main argument used in this paper is what could be called siso: symmetry in, symmetry out. This principle appears to be universal in mathematics: whenever that which enters a formal process (a proof, an algorithm, whatever) obeys a certain symmetry, what comes out obeys at least the same symmetry. For example, because $i^2 = -1$, but also $(-i)^2 = -1$, there is an automorphism of the complex plane called complex conjugation and all statements that do not include functions or constants that explicitly break this symmetry (such as the function that extracts the imaginary part, or the function that determines the argument of a complex number, or a non-real constant) are preserved if all occurrences of i are exchanged for $(-i)$.

As an application of the "siso principle" in this situation, consider the following way to prove that i^i must be a real number[18] without actually computing it: assume $i^i = a + bi$, with $a, b \in \mathbb{R}$. Then $(-i)^{-i} = a - bi$ because according to the siso principle, we may replace all i with $-i$. Using that $-i = i^{-1}$, we find that i^i and $(-i)^{-i}$ are equal: $(-i)^{-i} = (i^{-1})^{-i} = i^{-(-i)} = i^i$, and hence $a + bi = a - bi$, thus $b = 0$ and $i^i = a \in \mathbb{R}$.

Although this principle is a powerful tool of mathematical reasoning, I also view it as one of the fundamental limitations of the ability of mathematics to model reality. Failure to consider that reality is not necessarily bound by the same restrictions as its mathematical models has led to such bizarre ideas as Buridan's ass or the many-worlds interpretation of quantum mechanics[19].

[18] Let's put aside the issue of complex exponentiation being multi-branched. Or let's view this as a way to show that all the branches yield real-valued answers.

[19] Although the latter is apparently also based on a failure to understand the implications of the

8 Open Questions

Solving ooops seems conceptually related to strategies for cheating at multiple-choice questions that rely on the habit of test designers to provide possible responses that are wrong but similar to the correct response. The exact relationship between these two kinds of problem solving are beyond the scope of this paper, but it might be interesting to think deeper about this.

There ought to be a better algorithm for generating ooops than brute force checking the many subsets of the set of pre-ooops over Σ. For example, given a link constant[20] ooop O, for all m the graph that has an edge between each two objects of O that agree on m features is regular, and the edge sets of these graphs are pairwise disjoint, which restricts the possibilities.

The category theoretical properties of ooops may be worth studying. A function $m : \Sigma \to \Sigma'$ is a morphism from an ooop O over Σ to an ooop O' over Σ' iff $m(O) = O'$. This naturally generalizes the notions of automorphisms and isomorphisms of ooops defined earlier. It would be interesting to see if the category of ooops has products, coproducts, or other limits which may be used to construct complex ooops from simpler ones. It is perhaps also helpful to not only consider ooops, but all pre-ooops, and then find out what characterizes the subcategory of ooops.

References

[1] Martin Gardner and Tom Ransom. "Mathematical Games". In: *Scientific American* (1975).

[2] Martin Gardner and Dana Richards. *The colossal book of short puzzles and problems*. Norton, 2006.

[3] Tanya Khovanova. "Odd One Out". In: *arXiv preprint arXiv:1005.2700* (2010).

superposition principle for linear equations, namely that the multiverse would then be equivalent to a collection of non-interacting systems (the eigenmodes of the Hamiltonian), in each of which nothing ever happens.

[20] See subsection 5.2 on page 39