

# Using Formal Concept Analysis in Mathematical Discovery

Simon Colton and Daniel Wagner  
sgc, dwagner@doc.ic.ac.uk

Combined Reasoning Group  
Department of Computing  
Imperial College London  
<http://www.doc.ic.ac.uk/crg/>

**Abstract.** Formal concept analysis (FCA) comprises a set of powerful algorithms which can be used for data analysis and manipulation, and a set of visualisation tools which enable the discovery of meaningful relationships between attributes of the data. We explore the potential of combining FCA and mathematical discovery tools in order to better facilitate discovery tasks. In particular, we propose a novel lookup method for the Encyclopedia of Integer Sequences, and we show how conjectures from the Graffiti discovery program can be better understood using FCA visualisation tools. We argue that, not only can FCA tools greatly enhance the management and visualisation of mathematical knowledge, but they can also be used to drive exploratory processes.

**Keywords** Automated Theory Formation, Formal Concept Analysis, Knowledge Discovery.

## 1 Introduction

Formal Concept Analysis (FCA) consists of a set of well established techniques for the analysis and manipulation of data. FCA has a strong theoretical underpinning, efficient implementations of fast algorithms, and useful visualisation tools. There are strong links between FCA and machine learning, and the connection of both fields is an active area of research [8, 12, 13]. We concentrate here on the combination of FCA tools with systems developed to aid mathematical discovery. In particular, we are interested in addressing (i) whether FCA algorithms can be used to enhance the discovery process and (ii) whether FCA visualisation tools can enable better understanding of the discoveries made.

To facilitate this study, we have implemented ways to integrate two FCA tools with (a) a system that uses HR [4] and Maple [18] to discover graph theory conjectures in a manner similar to the Graffiti program [5], and (b) the Online Encyclopedia of Integer Sequences, which is a very important mathematical database. In the first case, we show that the large number of conjectures which are produced can be efficiently organised and better visualised using a lattice structure afforded by representing the conjectures as a formal context. In the second case, we show that the data manipulation aspects of FCA enable a new

way to mine information from the Encyclopedia. In particular, we describe a system we have implemented which can return sensible matches to query sequences that neither the Online Encyclopedia nor its more powerful sister program – the superseeker server – can explain.

This paper is organised as follows. In the next section, we briefly describe the theory, applications and implementations of Formal Concept Analysis. In Section 3, we describe the Graffiti program, our HR/Maple simulation of it and the Encyclopedia of Integer Sequences. In Section 4, we describe the methods we have developed in order to use FCA tools in conjunction with the HR/Maple simulation and the Encyclopedia. Following this, in Section 5 we describe some experiments with both the HR/Maple/FCA combination and the Encyclopedia/FCA combination. In particular, we look at the sensitivity and selectivity of the lookup method afforded by the Encyclopedia/FCA combination, and we give some examples of it in use. We also provide an illustrative example of using FCA visualisation tools to better understand a set of graph theory conjectures produced by the HR/Maple system. We conclude in Section 6 by suggesting that FCA visualisation and data analysis tools could be used not only to enhance mathematical discovery, but also to drive the discovery process.

## 2 Formal Concept Analysis

Formal concept analysis is a mathematical theory of conceptual hierarchies. A *formal context* is defined as a set of formal objects  $\mathcal{O}$ , a set of formal attributes  $\mathcal{A}$  and a relation  $I$  on  $\mathcal{O} \times \mathcal{A}$ . The relation  $oIa$  for  $o \in \mathcal{O}$  and  $a \in \mathcal{A}$  can be read as “object  $o$  has attribute  $a$ ” or “attribute  $a$  is true for object  $o$ ”. Such a binary context can be represented as a “cross table” (cf. Fig. 1). The set of common attributes of a set of objects  $O \in \mathcal{O}$  is defined as  $O' = \{a \in \mathcal{A}: oIa \forall o \in O\}$ . Analogously, the set of common objects of a set of attributes  $A \in \mathcal{A}$  is defined as  $A' = \{o \in \mathcal{O}: oIa \forall a \in A\}$ . A *formal concept* is a pair of sets  $(O, A)$  where  $O' = A$  and  $A' = O$ . The set  $O$  is called the *extent* and  $A$  the *intent* of the formal concept  $C = (O, A)$ . Formal concepts correspond to maximally filled rectangles in the cross table. Together with the subconcept relation, which is the sub-set (resp. super-set) relation on intents (resp. extents), a formal context forms a complete lattice. For more details on the mathematical theory of FCA, see [10]. For details of how FCA has been formalised in PVS and Mizar, see [11] and [16].

FCA has been successfully applied in various domains [9], and FCA theory is a good foundation for efficient algorithms as well as for human understandable reasoning. Visualisation of the concept lattice in what are known as *Hasse diagrams* is a particularly useful tool in human centred knowledge discovery. If the context is too large, its visualisation may become confusingly complex, but there are various methods in FCA to handle this complexity, e.g., conceptual scaling, nested line diagrams, product of sublattices [7, 10, 15]. To further clarify the diagrams, a reduced labeling is often used in FCA. Reduced labelings can be summarised as follows: (i) each node represents a formal concept (ii) objects are annotated below the most specific concept which contains them in its extent



### 3 Mathematical Discovery Systems

#### 3.1 The Graffiti Program simulated by HR and Maple

The Graffiti program [5] by Siemion Fajtlowicz makes conjectures of a numerical nature in graph theory. Given a set of well known, interesting graph theory invariants, such as the diameter, independence number, rank, and chromatic number, Graffiti uses a database of graphs to empirically check whether one sum of invariants is less than another sum of invariants. The empirical check is time consuming, so Graffiti employs two techniques, called the beagle and dalmation heuristics, to discard certain trivial or weak conjectures before the empirical test. If a conjecture passes the empirical test and Fajtlowicz cannot prove it easily, it is recorded in [6] and forwarded to interested graph theorists.

As an example, conjecture 18 in [6] states that, for any graph  $G$ :

$$cn(G) + r(G) \leq md(G) + fmd(G),$$

where  $cn(G)$  is the chromatic number of  $G$ ,  $r(G)$  is the radius of  $G$ ,  $md(G)$  is the maximum degree of  $G$  and  $fmd(G)$  is the frequency of the maximum degree of  $G$ . This conjecture was passed to some graph theorists, one of whom found a counterexample. The conjectures are useful because calculating invariants is often computationally expensive and bounds on invariants may bring computation time down. Moreover, these types of conjecture are of substantial interest to graph theorists, because they are simply stated, yet often provide a significant challenge to resolve – the mark of an important theorem such as Fermat’s Last. In terms of adding to mathematics, Graffiti has been extremely successful. The conjectures it has produced have attracted the attention of scores of mathematicians, including many luminaries from the world of graph theory. There are over 60 graph theory papers published which investigate Graffiti’s conjectures.

Unfortunately, Graffiti is not available for experimentation. However, in [14], we used a combination of the Maple computer algebra system and the HR automated theory formation system [4] to simulate Graffiti, and we produced very similar results. The way in which HR operates and the tools it has available for management of the mathematical information it produces have been described in [3]. A detailed description of how we used HR and Maple in graph theory is beyond the scope of this paper. However, we note that in order to further describe the conjectures produced, we calculated (i) a *tightness* measure which determined how close the inequality was to being equality (which is useful when using one summation of invariants to bound the calculation of another), and (ii) a slack constant for each conjecture which is the largest real number which can be added to the left hand side of the inequality without breaking it. For instance, the conjecture that for all graphs,  $G$ ,  $rank(G) \leq connectivity(G) + num\_vertices(G)$  has slack value 1, which means we can strengthen the conjecture to:  $rank(G) + 1 \leq connectivity(G) + num\_vertices(G)$ .

### 3.2 The Online Encyclopedia of Integer Sequences

The Online Encyclopedia of Integer Sequences,<sup>6</sup> contains more than 127,000 integer sequences, such as prime numbers, square numbers, the Fibonacci series, etc. They have been collected over 40 years by Neil Sloane, with contributions from hundreds (possibly thousands) of mathematicians. The Encyclopedia is very popular, receiving tens of thousands of queries every day. The first terms of each sequence are stored, and the user queries the database by providing the first terms of a sequence they wish to find a hit for. Sloane has recorded many times when using the Encyclopedia has led to a conjecture being made. For instance, in [17], he describes how a sequence that arose in connection with a quantization problem was linked via the Encyclopedia with a sequence that arose in the study of three-dimensional quasicrystals. Enabling such discoveries is one of the main purposes of the Online Encyclopedia. In addition to this web-service, there is also an email-service called the *superseeker*, which, as well as searching the Encyclopedia, performs extensive transformations on a given query sequence and on the Encyclopedia entries in order to find a match which explains the query.

## 4 Combining Mathematical Discovery Software and FCA

In the following sub-sections, we describe two applications where we have used FCA to (i) help visualise the results of a mathematical discovery system, and (ii) enhance the abilities of a mathematical discovery system.

### 4.1 Visualising Inequality Conjectures

A difficulty we had with the simulation of the Graffiti program described above was the sheer volume of the conjectures it produced. We ordered these in terms of the inequality tightness, but it was still difficult to pick out conjectures for certain graph invariants. Moreover, it was difficult to keep track of the chains of inequalities. For instance, it might look promising to investigate the conjecture that  $I_1 \leq I_2 + I_3$ , but this was made irrelevant by finding elsewhere in the list the stronger conjectures that, say,  $I_1 \leq I_{17}$  and  $I_{17} \leq I_2$ .

In order to better manage and visualise the inequality conjectures produced, we used the ConExp FCA system. To do so, after a session with HR/Maple simulating Graffiti, we extracted the set of summations,  $S$ , of invariants in the theory, and formed a binary context with them (note that  $S$  also contained the initial set of invariants). We used  $S$  both as the set of formal objects and the set of formal attributes in the binary context. Then, we extracted the set of inequalities produced and used the binary relation that an object  $s_o$  (summation of invariants) has attribute  $s_a$  (also a summation of invariants) if the conjecture  $x_o \leq s_a$  has been made. Deriving the formal context in this way enables us to read off inequality conjectures from the Hasse diagram: each node represents a

<sup>6</sup> <http://www.research.att.com/~njas/sequences>

summation of invariants or a set of summations, and if one node  $n_1$  is joined to another node,  $n_2$ , which is higher in the lattice, then the set of summations represented by  $n_1$  is conjectured to be less than or equal to the set of summations represented by  $n_2$ . We present a lattice for the first 120 conjectures produced by HR/Maple and we describe how the lattice size grows as the number of conjectures grows in Section 5.1.

## 4.2 Integer Sequence Lookup

As discussed above, the Online Encyclopedia of Integer Sequences is an extremely powerful and popular mathematics tool. However, there is some room for improvement in the way it searches for conjunctions of sequences. For instance, searching for this query sequence:  $1, 9, 36, 225$  returns a single hit, namely sequence A036907, which are square refactorable numbers (see [1]). However, searching for the sequence  $1, 36, 136, 276$ , which is in fact the first four *triangular* refactorable numbers returns no results. In the first case, the hit was successful, because someone had entered the concept of square refactorables into the Encyclopedia, but as no-one had done likewise for triangular refactorables, no hit was made in the second case.

We have implemented a routine which is able to efficiently construct such missing conjunctions of pairs, triples, quadruples, etc., of number types from the Encyclopedia. We start with the database,  $D$ , which was embedded in the NumbersWithNames program [2]. This is a snapshot of a fragment of the Encyclopedia taken in 2001, and contains 990 integer sequences which are of sufficient importance to have been given names, such as prime numbers, even numbers, pernicious numbers, etc.  $D$  contains only strictly increasing sequences, and represents roughly one hundredth of the current Encyclopedia entries. Given a query sequence  $Q = \{q_1, \dots, q_n\}$ , we first extract the set  $S = \{s \in D : \forall q_i \in Q, q_i \in s\}$ . This is effectively the set of database sequences which are supersequences of the query. We then set up a binary context with formal objects being the integers 1 to  $q_n$ , formal attributes being the set  $S$ , and the binary relation between objects and attributes expressed as the relationship of whether an integer (object) is part of a sequence (attribute). We then form a Hasse diagram with reduced labeling, and inspect the bottom node of the lattice. The intent of this node will be a conjunction of sequences in  $S$ , and is returned as the definition of a sequence which covers the query sequence. Note that the intent may include fewer terms than the set of supersequences of the query, due to the reduced labeling performed by FCA, yet the extent may contain more terms than the query sequence, as the intent may describe more than just the query terms.

As an illustrative example, take the sequence  $88, 124, 216, 246$ . Note that this returns no hit from the Online Encyclopedia, and even superseeker fails to find a way of explaining this sequence. Our system returns three hits from the NumbersWithNames database, and constructing the binary context as described above produces the lattice presented in Fig. 2. We see that, in this case, the intent covers perfectly the query sequence in its extent, i.e., our system has discovered that the query sequence can be described as the set of untouchable

Erdos-Woods numbers which are palindromic when written in base 5. While this is certainly not a simple definition, it is considerably easier to understand than many returned from the superseeker server. As another example, if we start with the query sequence  $12, 30, 42, 56$ , our system informs us that the sequence of heteromecic, semi-perfect, balanced numbers has this sequence:  $6, 12, 30, 42, 56$ , which is a super-sequence of our query sequence.

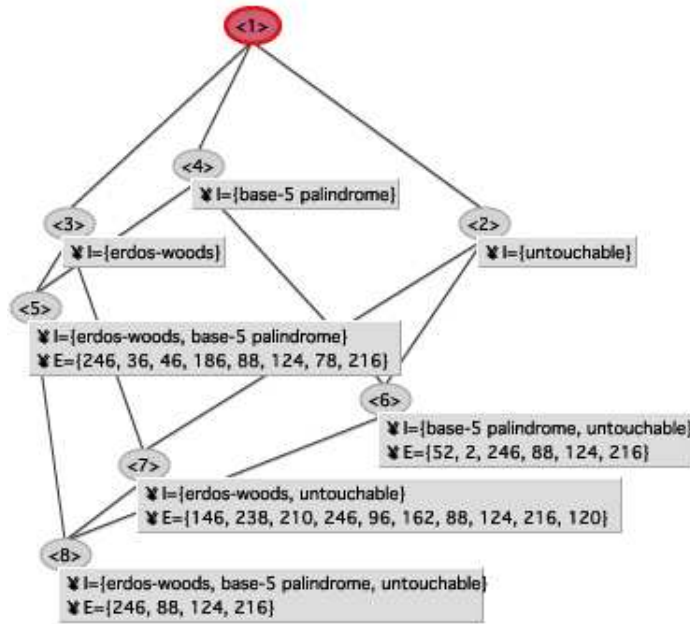


Fig. 2: Hasse diagram for sequence lookup (produced by Galicia).

Successful database lookup methods have to find a balance between sensitivity and selectivity. Sensitivity is the ability of the method to find genuine hits for a high proportion of queries it receives. Selectivity, on the other hand, is the ability of the method to avoid false positives, i.e., answers to queries which are not genuine hits. We have found that selectivity can be a problem for our method. For instance, if we start with the sequence  $2, 18, 68, 102, 116$ , then there are only two sequences from the database which contain all the query terms, namely A005843 (even numbers) and A045954 (even lucky numbers). In this case, therefore, our method will return the concept of integers which are both even and even lucky numbers as the hit for this sequence. The extent of this concept is, of course, the sequence of even lucky numbers:  $2, 4, 6, 10, 12, 18, 20, 22, 26, 34, 36, 42, 44, 50, 52, 54, 58, 68, 70, \dots$ . Clearly this is too general, and hence not a genuine answer to the query. Hence, to keep the selectivity high, such answers should not be output by our method.

Another way in which an answer might be perceived as being not genuine is if it is too specialised. For instance, given the sequence *5, 20, 23, 29*, left unchecked, our method returns the answer that these integers are congruent, undulating, fibonacci-lucky, evil, cube-free, babylonian, noncube, nonsquare, unlucky, arctangent irreducible/stormer, biquadratfree and weak numbers. This conjunction of 13 sequence definitions covers perfectly the integers 5, 20, 23, and 29, and no others between 1 and 29. However, the answer is hardly satisfying. We therefore need a way in which to constrain our method to output concepts which are not too specific in their intent, yet not too general in their extent.

To improve the selectivity of our approach, we use two constraints to rule out certain sequences from  $D$  from inclusion in the answer. Firstly, sequences must have a density less than 0.5 over the region of the number line they occupy (a sequence  $s_1, \dots, s_n$  has density  $\frac{n}{s_n - s_1}$ ). This rules out very general sequences such as square free integers. As roughly 6 in every 10 integers less than 100 are square-free, this concept is unlikely to form a property of a genuine hit. Secondly, we maintain a *black-list* of sequences which are found in answers too often (and hence are likely to be too general in their extent). We derived this list experimentally, by randomly generating 10 sequences, using our method to construct an answer, and then adding any sequence to the black-list if it appeared in three or more answers. We repeated this a number of times until no sequence appeared more than three times for a few turns.

The black-list currently contains the following sequences (with an asterix signifying a wild-card): nilpotent, nialpdrome\*, smooth\*, panconsummate, loeschian, odd, even, odd square free, even cototient, harshad/niven, equidigital, prime power, practical\*, digitised partition, cyclic, amino acid, contracted, power-sum, flimsy, higgs' prime. Note that some black-listed sequences have density greater than 0.5, and are hence caught by both constraints. We have kept them in the list for the purpose of experimentation (see Section 5.2). These two constraints very much reduce the set of database sequences which can be conjoined, and so the intent of an answer rarely contains more than a few sequences. However, we have a final filter on the output: if the intent contains a conjunction of more than five sequences, it is not shown to the user. We present some experiments with this lookup method in Section 5.2.

## 5 Experiments and Results

### 5.1 Graph theory Visualisation

For our experiments in graph theory, we used the HR/Maple simulation of Graffiti as described above to generate conjectures about inequalities of graph invariants. Given the background definitions *number of vertices*, *number of edges*, *diameter*, *connectivity*, *countcuts*, *max-degree*, *min-degree*, *counttrees*, *rank*, *average degree*, *average temperature*, *radius* and *chromatic number*, we ran HR for 10000 steps which produced 66 new summations of the given 13 graph invariants, and 820 inequality conjectures. Using subsets of these conjectures, we calculated the formal context with the graph invariants as objects and attributes and the

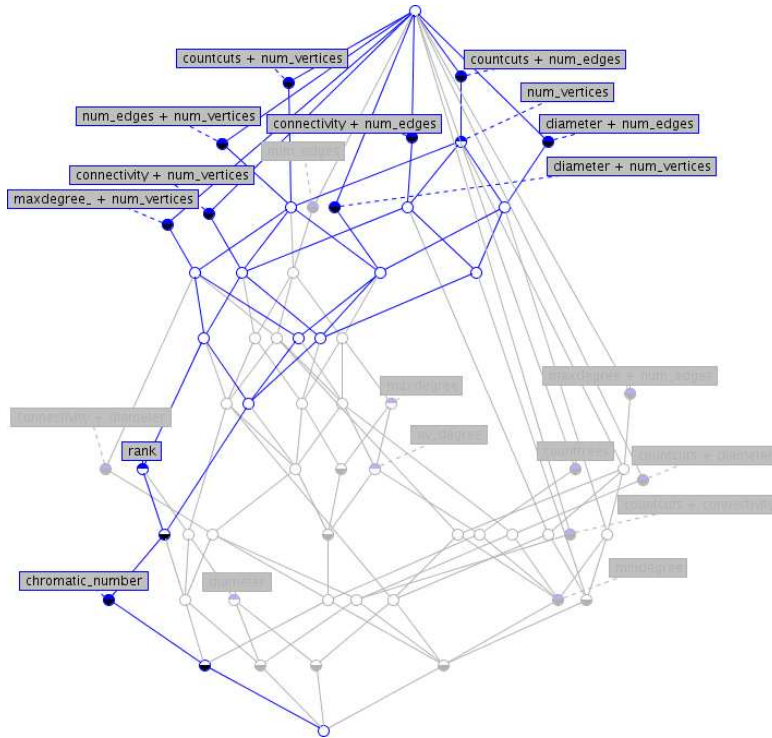


Fig. 3: Concept lattice for graph invariants, focused on chromatic numbers.

less than or equal to relation and then used ConExp to generate the Hasse diagram for inspection. For instance, in Fig. 3, we present the lattice derived from the first 120 conjectures that HR/Maple produced. We have used the ConExp interaction facilities to highlight the chromatic number node, which enables us to read off the conjectures involving that invariant. For comparison, in table 1, we show the kind of output that HR/Maple is able to produce, namely the top 10 conjectures from the 120 conjectures when they are ordered in decreasing tightness of the inequality in the conjecture. Compared to just a list of the inequality conjectures, we see that the FCA lattice enables us to much more easily read chains of inequalities. For instance, we can clearly read the chain of inequalities:  $chromatic\_number \leq rank \leq max\_degree + num\_vertices$ .

Given the highlighting abilities that ConExp has, we are able to mine information easily from more complex lattices. However, it is informative to look at how complex the lattices become as the number of conjectures they are used to represent grows. In Fig. 4, we plot the number of nodes in the lattice versus the number of inequality conjectures. We see that up to around 400 conjectures, the lattice contains fewer nodes than conjectures. However, after this point, the number of nodes rises quite sharply. Indeed, after 600 conjectures, there is no gain in clarity from using FCA because there are more nodes than conjectures.

Tightness	Slack	Conjecture
0.998	0	$connectivity(G) \leq mindegree(G)$
0.918	0	$rank(G) \leq num\_vertices(G)$
0.842	0	$connectivity(G) + diameter(G) \leq num\_vertices(G)$
0.823	0	$av\_degree(G) \leq max\_degree(G)$
0.803	1	$max\_degree(G) \leq num\_vertices(G)$
0.799	0	$radius(G) \leq diameter(G)$
0.793	1	$rank(G) \leq connectivity(G) + num\_vertices(G)$
0.781	0	$chromatic\_number(G) \leq rank(G)$
0.769	0	$min\_degree(G) \leq av\_degree(G)$
0.767	0	$connectivity(G) \leq av\_degree(G)$

Table 1: HR/Maple’s tightest inequalities (from 120 conjectures in graph theory).

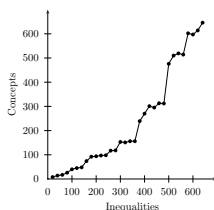


Fig. 4: Number of inequalities versus number of formal concepts.

## 5.2 Encyclopedia of Integer Sequences Lookup

Initial experiments with the integer sequence lookup approach described in Section 4.2 showed much promise. To begin to determine whether the method could be used in practice (perhaps as part of the superseeker server), we need to test three aspects: (i) lookup speed (ii) sensitivity (iii) selectivity. The first of these is easy to deal with - the time taken to generate an answer to a query has been around 40 milliseconds, which would probably make it fast enough for a server.

Assessing the sensitivity of our method is somewhat problematic, as we need to assess the probability of a suitable answer being returned for any reasonable query. Such a measure of sensitivity will become clear as the lookup service is used for real queries. However, it is clear that, if the user submits a sequence which is a perfect conjunction of two to five number types in the Numbers-WithNames database, then the lookup method will return an answer. The only exceptions to this would be if the number types were too dense on the number line, or if one of the sequences was black-listed, and both these scenarios are unlikely for reasonable query sequences. A more tractable question involves the selectivity of our method – if it consistently returns spurious answers to queries, then it will be de-valued and not used. To address selectivity, we ran three sets of 11 experiments, with the results presented in table 2.

In experiment 1a, we used the lookup method with the maximum density, black-list and maximum definition constraints all imposed. We generated ran-

experiment	Constraints			Random		Extent				Intent			Speed
	density	black-list	def-limit	random add	random tried	av. seq. length	av. hit length	perfect hits	contig. hits	av. super	av. defs	biggest. def	av. time (ms)
1a	0.5	yes	yes	25	129845	4.117	8.157	73	168	2.434	2.323	5	20.665
2a	0.5	yes	no	25	131046	4.112	8.147	72	138	2.447	2.368	10	19.491
3a	0.5	no	yes	25	9943	4.396	16.161	8	40	2.812	2.707	5	15.656
4a	0.5	no	no	25	9048	4.4	15.878	26	53	3.216	3.034	13	14.933
5a	1.0	yes	yes	25	2281	5.477	38.186	1	1	4.863	3.477	5	33.973
6a	1.0	yes	no	25	1405	5.138	30.318	4	10	6.84	5.283	13	34.594
7a	1.0	no	yes	25	2603	5.641	37.519	0	0	4.801	3.46	5	24.949
8a	1.0	no	no	25	1341	5.19	27.56	13	29	7.091	5.512	18	32.153
9a	0.5	no	no	25	-	4.117	6.946	158	314	6.354	5.679	15	18.602
10a	1.0	yes	no	25	-	4.117	5.849	293	360	11.009	8.553	16	42.948
11a	1.0	no	no	25	-	4.117	5.285	440	512	15.12	11.512	23	83.639
1b	0.5	yes	yes	50	253540	4.09	12.566	25	47	2.274	2.234	5	29.786
9b	0.5	no	no	50	-	4.09	10.936	62	112	4.777	4.589	15	33.631
10b	1.0	yes	no	50	-	4.09	10.456	88	117	6.123	5.272	15	61.951
11b	1.0	no	no	50	-	4.09	9.067	159	213	8.786	7.406	21	50.171
1c	0.5	yes	yes	100	829628	4.065	17.851	7	20	2.377	2.368	5	45.432
9c	0.5	no	no	100	-	4.065	16.048	24	58	3.956	3.904	14	32.083
10c	1.0	yes	no	100	-	4.065	16.944	23	36	3.524	3.298	15	37.392
11c	1.0	no	no	100	-	4.065	15.242	58	88	5.182	4.782	21	37.305

Table 2: Sequence lookup for 1000 randomly generated sequences.

dom sequences in the following way: we generated a random integer between 1 and 25 as the first entry in the sequence, then added this to another random integer between 1 and 25 to get the second entry, and continued in this way until our sequence contained between 4 and 7 (inclusive) integers, with the length also determined randomly. This gives strictly increasing sequences with integers roughly between 1 and 100 on the number line, which is the kind of query sequence we may expect (with perhaps a bias towards smaller numbers than for usual queries). As the method with all three constraints is particularly selective, it only returned hits for 1000 out of 129845 randomly generated sequences.

We recorded a number of statistics about these 1000 sequences. Firstly, we compared the length of the query sequence with the length of the sequence which was returned (i.e., the extent of the conjunction of sequence definitions). A large difference in the two lengths means that the sequence returned is unlikely to be a genuine hit for the sequence. In experiment 1a, the average sequence length was 4.1, and the average hit length was 8.157, hence we could expect approximately double the integers in the hit than the query. However, for 73 of the 1000 sequences, a perfect hit was returned, i.e., the hit was exactly the query sequence. Also, 168 of the returned sequences contained the query sequence contiguously, i.e., the hit was either perfect or had some initial integers which were not part of the query sequence (our method precludes trailing integers). Note that this kind of hit is also returned by the Encyclopedia of Integer Sequences, as the initial terms of a query sequence are often omitted.

In addition to details of the extent of the concept returned, we also recorded details of the intent returned. In particular, we recorded the average number of sequence definitions which were conjoined in the definition of the hit. In experiment 1a, the average was 2.323. Note that the average number of super-sequences of the query sequence was 2.434. Hence, we see that the FCA technique

Sequence	Description
6,30,36,60	eban & evil & pseudo-perfect/semi-perfect & highly abundant
2,11,23,48	semi-fibonacci & problime(3)
13,31,41,53,71	prime & primitive congruent & regular prime & short-period prime
13,37,41,53	prime & primitive congruent & gaussian prime & short-period prime
4,25,38,58,74	fibonacci-lucky & semi-prime/2-almost prime & twin fibonacci-lucky(1)
1,8,22,34	multiplicatively perfect & polyomino
8,21,32,50	arc-cotangent reducible/non-stormer & duffinian
3,18,20,30	evil & base-4 colombian/self & highly abundant
24,30,43,53	evil & strict egyptian(1)
3,17,21,38	base-4 palindrome & arc-cotangent reducible/non-stormer
6,15,23,30	evil & binary colombian/self & primitive congruent
2,5,25,38,59	fibonacci-lucky & base-4 palindrome & twin fibonacci-lucky(1)
15,23,48,51	rhombic & evil & binary colombian/self
4,6,30,46	eban & 2-knodel & binary colombian/self
6,12,27,36	evil & truncated triangular(1)
6,18,36,54	evil & pseudo-perfect/semi-perfect & base-8 palindrome
15,23,43,48	rhombic & evil & problime(3)
1,4,25,32	fibonacci-lucky & base-7 armstrong(2) & perfect power
7,19,23,32	rhombic & fibonacci-lucky & friendly/happy
2,7,13,37	prime & fibonacci-lucky & exceptional prime & class-1/pierpoint prime & absolute prime
1,10,20,26	semimorphic & base-3 palindrome & davenport-schinzel(1)

Table 3: Perfect hits from experiment 1a which are missing from the Encyclopedia.

of reducing the intent to remove redundant definitions is effective, even when there are only a few definitions conjoined (this effect becomes greater in later experiments, e.g., experiment 11a, where the reduction is by between 3 and 4 terms on average). In summary, if our method does return an answer, then there is a roughly 1 in 10 chance that it will be high quality, i.e., perfect and/or contiguous. Moreover, the definition of the hit will be fairly understandable – on average a simple conjunction of either two or three sequence definitions.

It is difficult to know in advance whether this will be selective enough for users of the lookup method. However, we can show that our method could be much less selective. In experiments 2a to 8a, we varied the usage of the definition length, density and black-list constraints. As table 2 shows, with the exception of removing only the definition length constraint, the method is far less selective when we remove the constraints. For instance, in experiment 8a, there were no constraints used. The method returned an answer for most of the query sequences (the exceptions were queries able to be answered with only one sequence – which were still ruled out). However, the answers returned were very low quality. On average, the answer to a query contained 22 more terms than the query – effectively making them useless answers – and the method found only 13 perfect hits. Moreover, the definitions of the answers were more complicated, being a conjunction of on average 5.5 sequence definitions.

In experiments 9a, 10a and 11a, we tested how the unconstrained methods perform on the sequences which passed the selectivity test from experiment 1, i.e., the 1000 sequences which returned a hit from the constrained method. The results from these tests were very interesting: it appears that if a query sequence does have some semantic value with respect to the number types in the database (i.e., should have a description according to the selectivity criteria), then the unconstrained lookup methods can often synthesise a more complicated but more accurate answer. Experiment 9a is particularly interesting: here the returned solutions were on average conjunctions of only 6.354 sequence definitions, but they included 158 perfect hits, which was more than double than in experiment 1a. In experiment 11a, the accuracy of the hits was striking: in 440 cases, the method returned a perfect hit. However, the cost for this was more complicated definitions of the hits: on average they were conjunctions of 11.512 sequences. This suggests a two-tier approach for the lookup method: (i) filter the sequence using the selectivity criteria, and return nothing if it fails the test (ii) for any sequence which passes the test, return both the answer from the constrained lookup and from the unconstrained lookup. The former answer will be easier to understand but perhaps less accurate, while the latter answer will be more difficult to understand, but is likely to be more accurate.

We repeated the set of 11 experiments twice. In the second set, we used 50 rather than 25 as the interval with which to generate random sequences, and in the third set we used 100. We report only the results from the first experiment in the set (with all the constraints imposed), and the final three experiments (with fewer constraints, applied to the sequences passing the selectivity criteria of the first experiment). We observe that finding solutions becomes increasingly difficult as the sequences spread out over the number line, which was a trend we expected. However, we didn't expect the trend that the number of definitions in the unconstrained answers reduces as the sequences spread out. For instance, in the first set of experiments, the unconstrained method produced around 6 times more perfect hits than the constrained method, but needed to conjoin around 4 times the number of definitions. In contrast, in the third set of experiments, the unconstrained lookup method found more than 8 times more perfect hits, but used only twice the number of definitions. While this phenomenon is explainable – as fewer sequences from the Encyclopedia will match sparse sequences – it adds weight to our proposal of using a two-tiered approach to sequence lookup.

In table 3, we list the 21 sequences for which our fully constrained method (experiment 1a) returned a perfect hit, whereas the Encyclopedia of Integer Sequences returned no hits. The average number of conjoined definitions for these sequences is 2.95, which is higher than the average for experiment 1a, suggesting that sequences have to be more complex to be missing from the Encyclopedia.

## 6 Conclusions and Future Work

Formal Concept Analysis is a well developed area with much to offer for data analysis in various application domains. We have investigated the usage of FCA

for analysis, manipulation and visualisation of mathematical knowledge/data. In particular, we have addressed the question of whether FCA could be used to enhance systems used for automated or semi-automated mathematical discovery. In the application to database lookup from the Encyclopedia of Integer Sequences, we have shown that FCA tools are useful for manipulation of mathematical information, and our system was able to find sensible answers to numerous query sequences which the Encyclopedia (and in some cases superseeker) couldn't answer. In the application to visualising the graph theory conjectures produced by our HR/Maple system, we showed that FCA visualisation tools have much potential for better management of machine generated mathematics.

In a further set of experiments which there hasn't been space to describe here, we have used the implication generation tools within ConExp with the results from using HR in number theory. In a particular test, we wanted to see whether using HR followed by FCA concept exploration could help us find a particular conjecture quicker than using HR alone. Starting with just the ability to multiply two numbers, HR can discover the conjecture that odd refactorable<sup>7</sup> numbers are squares. With fairly strict restrictions on the search that HR can perform, it still takes 721 steps to produce 1883 conjectures, the last of which is the one we wanted. However, if we stop HR after only 64 steps, and then use ConExp to generate all implication conjectures possible from the 22 concepts that HR has produced, the conjecture that we want is output. We need to perform further tests, but it seems likely that a permanent combination of HR with an FCA system could dramatically reduce the combinatorial burden that HR has when making conjectures, and FCA could improve HR's discovery process as a whole.

This not only strengthens our claim that the combination of FCA tools with discovery systems has much potential to enhance discovery, but it also suggests a more fine-grained involvement of FCA in discovery tasks. In particular, in future work, we plan to build a hybrid FCA/machine learning system which is of benefit to both the machine learning and the FCA communities. Among other benefits, we intend the system to improve upon (a) FCA systems, by using concept formation abilities similar to those from machine learning, and (b) the visualisation and user-interaction abilities of machine learning systems. The system will enable FCA tools to drive the exploration process using machine learning enhancements. We hope to show that the hybrid system enables users to make more interesting discoveries in mathematical domains than they would using FCA or machine learning tools alone. We also intend to apply the hybrid system to discovery tasks in other domains, such as bioinformatics.

## Acknowledgements

We would like to thank the anonymous reviewers for their useful comments, and also Bernhard Ganter for very helpful discussions on Formal Concept Analysis.

---

<sup>7</sup> Refactorable numbers,  $n$ , are such that the number of divisors of  $n$  is itself a divisor.

## References

1. S Colton. Refactorable numbers - a machine invention. *Journal of Integer Sequences*, 2, 1999.
2. S Colton and L Dennis. The numberswithnames program. In *Proceedings of the Seventh AI and Maths Symposium*, 2002.
3. S Colton, P Torres, P Cairns, and V Sorge. Managing automatically formed mathematical theories. In *Proceedings of the 5th International Conference on Mathematical Knowledge Management*, 2006.
4. Simon Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.
5. S Fajtlowicz. On conjectures of Graffiti. *Discrete Mathematics* 72, 23:113–118, 1988.
6. S Fajtlowicz. The writing on the wall. Unpublished preprint, available from <http://math.uh.edu/~clarson/>, 1999.
7. Bernhard Ganter. *Formal Concept Analysis. Foundations and Applications*, chapter Contextual Attribute Logic of Many-Valued Attributes, pages 101–113. Springer-Verlag, 2005.
8. Bernhard Ganter and Sergei O. Kuznetsov. Hypotheses and version spaces. In A. de Moor, W. Lex, and B. Ganter, editors, *International Conference on Conceptual Structures*, volume 2746 of *Lecture Notes in Artificial Intelligence*, pages 83–95. Springer-Verlag, 2003.
9. Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors. *Formal Concept Analysis. Foundations and Applications*. Springer-Verlag, 2005.
10. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.
11. Maria Jose Hidalgo Francisco Jesus Martin-Mateos Jose Luis Ruiz-Reina Jose Antonio Alonso, Joaquin Borrego. Verification of the Formal Concept Analysis.
12. Sergei O. Kuznetsov. Machine learning and formal concept analysis. In P. Euklund, editor, *International Conference on Formal Concept Analysis*, number 2961 in *Lecture Notes in Artificial Intelligence*, pages 287–312. Springer-Verlag, 2004.
13. Michel Liquiere and Jean Sallantin. Structural machine learning with galois lattice and graphs. In *International Conference on Machine Learning*, 1998.
14. K Mohamadali. A rational reconstruction of Graffiti. Master’s thesis, Department of Computing, Imperial College, London, 2003.
15. Patrick Scheich, Martin Skorsky, Frank Vogt, Cornelia Wachter, and Rudolf Wille. *Information and Classification - Concepts, Methods and Applications*, chapter Conceptual Data Systems, pages 72–84. Springer-Verlag, 1992.
16. Christoph Schwarzweller. Mizar formalization of concept lattices.
17. N J A Sloane. My favorite integer sequences. In *Proceedings of the International Conference on Sequences and Applications*, 1998.
18. Waterloo Maple. *Maple Manual at <http://www.maplesoft.on.ca>*.
19. Serhiy A. Yevtushenko. System of data analysis ”concept explorer”. In *Proceedings of the 7th national conference on Artificial Intelligence KII*, pages 127–134, 2000.