Initial Experiments with Statistical Conjecturing over Large Formal Corpora

Thibault Gauthier[1], Cezary Kaliszyk[1], and Josef Urban[2]

[1] University of Innsbruck, Austria
{thibault.gauthier,cezary.kaliszyk}@uibk.ac.at
[2] Czech Technical University in Prague
josef.urban@gmail.com

Abstract. A critical part of mathematicians' work is the process of conjecture-making. This involves observing patterns in and between mathematical objects, and transforming such patterns into conjectures that describe or better explain the behavior of the objects. Computer scientists have since long tried to reproduce this process automatically, but most of the methods were typically restricted to specific domains or based on brute-force enumeration methods. In this work we propose and implement methods for generating conjectures by using statistical analogies extracted from large formal libraries, and provide their initial evaluation.

1 Introduction

In the past decade, there has been considerable progress in proving conjectures over large formal corpora such as Flyspeck [7], Isabelle/HOL [12], the Mizar Mathematical Library (MML) [1] and others. This has been achieved by combining high-level learning or heuristic fact-selection mechanisms with a number of methods and strategies for guiding the strongest (typically superposition-based) automated theorem provers (ATPs). While this approach has not reached its limits [2], and its results are already very useful, it still seems quite far from the way humans do mathematics. In particular, even with very precise premise (fact) selection, today's ATPs have trouble finding many longer Mizar and Flyspeck proofs of the toplevel lemmas in their libraries. In fact, only a few of such lemmas would be called a "theorem" by a mathematician. Often the "theorem" would be just the final proposition in a formal article, and the toplevel lemmas preceding the theorem would be classified as simple technical steps that sometimes are not even mentioned in informal proofs.

An important inductive method that mathematicians use for proving hard problems is *conjecturing*, i.e., proposing plausible lemmas that could be useful for proving a hard problem.[3] There are likely a number of complicated inductive-deductive feedback loops involved in this process that will need to be explored, however it seems that a major inductive method is *analogy*. In a very high-level

[3] A famous example is the Taniyama-Shimura conjecture whose proof by Wiles finished the proof of Fermat's Last Theorem.

way this could be stated as: *"Problems and theories with similar properties are likely to have similar proof ideas and lemmas."*

Again, analogies can likely be very abstract and advanced. Two analogue objects may be related through a morphism. They can also be two instances of the same algebraic structure. The organization of such structures was recently addressed by the MMT framework [13] in order to represent different logics in a single consistent library. In this work we start experimenting with the analogies provided by *statistical concept matching* [4]. This method has been recently developed in order to align formal libraries of different systems and to transfer lemmas between them [5]. Here we use statistical concept matching to find the *most similar sets of concepts* inside one large library. The discovered sets of matching concepts can then be used to translate a hard conjecture C into a "related" conjecture C', whose (possible) proof might provide a further guidance for proving C. The remaining components that we use for this first experiment are standard large-theory reasoning components, such as fast machine-learners that learn from the thousands of proofs and propose the most plausible lemmas for proving the related conjectures, and first-order ATPs – in this case we use Vampire 4.0 [10].

2 Matching concepts

In order to apply some analogies, we first need to discover what they are by finding similar concepts. For our initial evaluation, our similarity matching will be limited to concepts represented by constants and ground sub-terms. Later this can be extended to more complex term structures. We describe here a general concept matching algorithm inspired and improved upon [4] and discuss how this algorithm can be adapted to find analogies within one library.

2.1 Matching concepts between two libraries

Given two theorem libraries, the first step is to normalize all statements, as this increases the likelihood of finding similar theorem shapes. If two theorems have the same shape (that we will call such abstract shapes a property), then the concrete constants appearing in this two theorems at the same positions are more likely to be similar. We will say that such pairs of constants are derived from the theorem pair.

Example 1. Given the theorems T_1 and T_2 with the statements, their respective normalizations, and the properties extracted from their statements:

$$T_1 :\ \forall x : num.\ x = x + 0 \qquad T_2 :\ \forall x : real.\ x = x \times 1$$
$$P_1 : \lambda num, +, 0.\ \forall x : num\ x = x + 0 \qquad P_2 : \lambda real, \times, 1.\ \forall x : real.\ x = x \times 1$$

The properties P_1 and P_2 are α-equivalent, therefore the theorems T_1 and T_2 form a matching pair of theorems, and the following three matching pairs of constants are derived:

$$num \leftrightarrow real,\ + \leftrightarrow \times,\ 0 \leftrightarrow 1$$

We further observe that the matchings $(+, \times)$ and $(0, 1)$ are in relation through the theorem pair (T_1, T_2). The strength of this relation – *correlation* – is given by the number and accuracy of the theorem pairs these matchings are jointly derived from. We will call the graph representing the correlations between different matchings the *dependency graph*. The *similarity score* of each matching (i.e., pair of concepts) is then initialized with a fixed starting value and computed by a dynamical process that iterates over the dependency graph until a fixpoint is reached.

The principal advantage of this method is that the algorithm is not greedy, allowing a concept to match multiple other concepts in the other libraries. This is a desirable property, since we want to be able to create multiple analogues for one theorem. Matchings are then combined into *substitutions*, which are in turn applied to the existing theorem statements yielding plausible conjectures.

Very few adjustments are needed to adapt this method to a single library. We create a duplicate of the library and match it with its copy. Since we are not interested in identity substitutions, we prevent theorems from matching with their copies. However, we keep self matches between constants in the dependency graph since good analogies can be often found by using partial substitutions.

3 Context-dependent substitutions

Constant matchings by themselves create special one-element substitutions that transport the properties of one constant to another. In general, substitutions are created from translating more than one constant. Suppose, that we know that addition is similar to union and multiplication to intersection. We can hope to transport the distributivity of multiplication over addition to the distributivity of intersection over union. Therefore, the correlations between the matchings are crucial for building the substitutions.

We now present two methods for creating substitutions from a theorem. These methods are based on the correlations between the concept pairs and the similarity score of each concept pair.

We want to construct substitutions that are most relevant in the local context, i.e., for the symbols that are present in the theorem we want to translate (*target theorem*).

The first method starts by reducing the dependency graph to the subgraph of all concept pairs whose first element is contained in the target theorem. We first select a starting node in this subgraph which is not an identity matching, and recursively find nodes (possibly identities) that are connected by the strongest edges of the subgraph, under the constraint that no two selected nodes have the same first element. The algorithm stops when no new node can be added. The final set of nodes obtained in this way forms a partial substitution. We run this algorithm either for all starting nodes, or for those with similarity scores above a certain threshold. This produces a set of substitutions, which are effectively the most relevant substitutions for the target theorem. This process seems to produce many substitutions, however in practice, many of them are identical, which limits their total number.

The second method is a brute-force approach where we first find the set of concepts ($Matched(T)$) that match at least one concept in the set of concepts ($Concepts(T)$) present in the target theorem T. We would like to create all possible substitutions between $Concepts(T)$ and $Matched(T)$ and rank them, however this would often blow up the generation phase. To limit the number of possible substitutions, we remove possible matches by iterating the following process until the number of substitutions is below a chosen threshold (1000): We select the constant C in T with most remaining matchings, remove its worst match, recompute the number of remaining matches for all constants in T, and check if the number of substitutions is already below the threshold. If so, the process terminates, otherwise we continue the iteration.

Next, we select the 200 substitutions with the highest *combined score*. The combined score of a substitution S is computed by multiplying the average correlation and similarity in S, i.e. formally as follows:

$$CombinedScore(S) = AverageCorrelation(S) \times AverageSimilarity(S)$$

$$AverageCorrelation(S) = \frac{1}{|S|^2} \times \sum_{M \in S, M' \in S} Correlation(M, M')$$
$$AverageSimilarity(S) = \frac{1}{|S|} \times \sum_{M \in S} SimilarityScore(M)$$

On top of these combined scores, the diversity of substitutions can be maximized by the following shuffling. The process iteratively chooses a (not yet selected) substitution with the best *diversity score* and increases the set of selected substitutions $\mathfrak{T}$. These scores are then updated to penalize the substitutions that have more matchings in common with the already selected ones.

$$CommonMatchings(S,T) = |S \cap T|$$
$$DiversityScore(S) = \frac{CombinedScore(S)}{(1 + \sum_{T \in \mathfrak{T}} CommonMatchings(S,T))^3}$$

These substitutions will eventually be applied to the initial theorem to create new conjectures. We hope that if such conjectures can be proved, they will improve the AI/ATP methods by enriching the theory. We consider in Section 4 two possible scenarios where the combination of conjecturing by analogies and premise selection could be useful.

4 Scenarios

We will consider two scenarios for the use of conjecturing: without and with a user given goal.

In the first scenario no goal set by the user, and our system should decide what is the next step in the development of the whole library. In this context, our algorithm produces the most likely conjectures by applying the first substitution generation method on all theorems. We then evaluate the conjectures by trying to prove them. We estimate the difficulty of a proved conjecture by the number of lemmas that were used in its proof. We estimate the relevance of a conjecture by how much adding this conjecture as a new theorem in the library helps to automatically prove formalized statements appearing after it in the library.

In the second scenario, a goal is given and the system should figure a way how to prove it. The typical AI-ATP method is to search through all lemmas (in an evaluation this means the lemmas which occur before the goal), and select the most relevant ones using machine learning techniques. If however an important lemma was not proven or proven after the goal, the premise selection fails to produce it. Therefore, we propose a way to produce some of such relevant missing lemmas. This method is depicted in Fig 1. We first select the 20 best scoring substitutions for this goal, which creates 20 new goals. We then try to prove them using the AI-ATP system. If some of them are successfully proved, we obtain small sets of lemmas which were sufficient to prove the translated goals. These lemmas are then translated back to the original concepts. We run our AI-ATP system again and remove those it cannot prove. The final step is to try to prove the user goal from those additional lemmas. This strategy simulates the following thought process a human could have: "Can I prove this conjecture for a similar concept"?, "If so, what where the necessary lemmas?" "If some of this lemmas holds for my original concept, they would likely help me in my original proof".

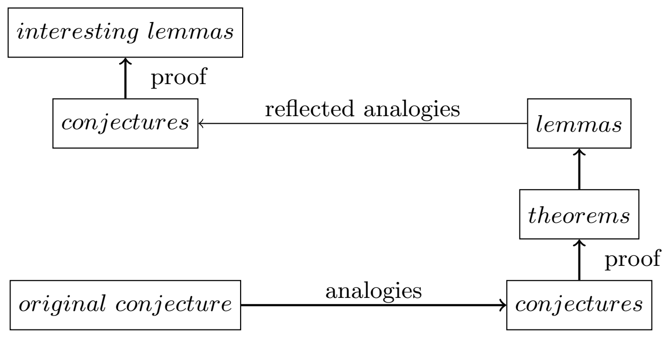

Fig. 1. Creating additional relevant lemmas for a conjecture through analogies.

5 Experiments

5.1 Untargeted Conjecture Generation

In the first scenario, we apply[4] the 20 "most plausible" substitutions to all MML theorems, and attempt to prove a small part (73535) of those, each time using the whole MML. We want to see how complicated and interesting the conjectures can be. After premise selection Vampire can prove 10% (7346) of them in 10 s, which is reduced to 4464 after pseudo-minimization [8] and removing tautologies and simple consequences of single lemmas. An example of a reasonably interesting conjecture (and analogy) with a new nontrivial proof is the convexity of empty subsets of real linear spaces, induced from a similar claim about them being "circled":[5]

```
registration
  let X be non empty RLSStruct;
  cluster empty -> circled for Element of bool the carrier of X;
```

Here "circled" is defined as[6]

```
definition
  let X be non empty RLSStruct; let A be Subset of X;
  attr A is circled means :Def6: :: RLTOPSP1:def 6
  for r being Real st abs r <= 1 holds r * A c= A;
```

and "convex" as[7]

```
definition
  let V be non empty RLSStruct; let M be Subset of V;
  attr M is convex means :Def2: :: CONVEX1:def 2
  for u, v being VECTOR of V
    for r being Real st 0 < r & r < 1 & u in M & v in M holds
      (r * u) + ((1 - r) * v) in M;
```

For example the following properties of circled[8] and convex[9] subsets are quite similar, leading the conjecturer into conjecturing further properties like the one stated above.

```
registration
  let X be RealLinearSpace;
  let A, B be circled Subset of X;
  cluster A + B -> circled ;
```

[4] The experimental version of our conjecturer used here is available at: http://147.32.69.25/~mptp/conj/conjecturing_standalone.tar.gz .
[5] http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#CC1
[6] http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#V3
[7] http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/convex1.html#V1
[8] http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rltopsp1.html#FC3
[9] http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/convex1.html#T2

```
theorem :: CONVEX1:2
 for V being non empty Abelian add-associative vector-distributive
   scalar-distributive scalar-associative scalar-unital RLSStruct
 for M, N being Subset of V st M is convex & N is convex holds
   M + N is convex
```

5.2 Targeted Conjecture Generation

In the second experiment, we have used as our target the set of 22069 *ATP-hard* Mizar toplevel lemmas (theorems). These are the theorems that could not be proved in any way (neither with human-advised premises, nor with learned premise selection) in our recent extensive experiments with state-of-the-art AI/ATP methods over the MML [9]. For the current experiment, those experiments are very thorough. They used high ATP time limits, many ATPs, their targeted strategies, a number of learning methods and their combinations, and a number of iterations of the learning/proving loop, approaching in total a month of a server time. Proving these problems with a low time limit and a single AI/ATP method is thus quite unlikely.

To each such hard theorem T we apply the 20 best (according to the diversity score) substitutions. Such substitutions are additionally constrained to target only the part of the MML that existed before T was stated. This gives rise to 441242 conjectures, which we attempt to prove – again only with the use of the MML that precedes T. Because of resource limits, we use only one AI/ATP method: k-NN with 128 best premises, followed by Vampire using 8 s. This takes about 14 hours on a 64-CPU server, proving 9747 (i.e. 2.2%) of the conjectures. We do two rounds of pseudo-minimization and remove tautologies and simple consequences of single lemmas. This leaves 3414 proved conjectures, originating from 1650 hard theorems, i.e., each such conjecture C is a translation of some hard theorem T under some plausible substitution σ ($C = \sigma(T)$). We translate the MML lemmas L^i_C needed for the proof of C "back" into the "terminology of T" by applying to them the reverse substitution σ^{-1}.

This results in 26770 back-translated conjectures. For each of them we hope that (i) it will be provable from the MML preceding T, and (ii) it will be useful for proving its T, since its image under σ was useful for proving $\sigma(T)$. We use again only 8 s to select those that satisfy (i), yielding after the minimization and removal of trivial ones 2170 proved back-translated lemmas, originating from 500 hard theorems. For each of these 500 theorems T we do standard premise selection (using slices of size 128 and 64) over the preceding MML, and add these lemmas (obviously only those that "belong" to T) to the premises of T. Then we run Vampire for 30 s on the standard and lemma-augmented problems. While there is no difference when using 128 lemmas, for 64 lemmas we obtain (in addition to the 6 proofs that both the standard and augmented methods find) an interesting new proof of the theorem `MATHMORP:25`[10], which cannot be found by Vampire using the standard method even with a time limit of 3600 s.

[10] `http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#T25`

```
theorem :: MATHMORP:25
for T being non empty right_complementable Abelian
            add-associative right_zeroed RLSStruct
for X, Y, Z being Subset of T
  holds X (+) (Y (-) Z) c= (X (+) Y) (-) Z
```

To find this proof, our concept matcher first used the statistical analogy between addition[11] and subtraction[12] in additive structures (`addMagma`). By doing that, it inadvertently constructed as a conjecture the theorem `MATHMORP:26`[13], that actually immediately succeeds `MATHMORP:25` in MML. This alone is remarkable, because this theorem was not known at the point when `MATHMORP:25` was being proved. Using premise selection and Vampire, `MATHMORP:26` was proved in 4 s, and a particular back-translated lemma from its proof turned out to be provable and crucial for proving `MATHMORP:25` automatically. This lemma is actually "trivial" for Mizar, since it follows by climbing Mizar's extensive type hierarchy [6] from an existing typing of the `''(-)''` function. However, as mentioned above, we were not able to get this proof without this lemma even with much higher time limits.

6 Conclusion and Future Work

We have investigated the application of a concept matching algorithm to formulate conjectures through analogies. We have described a way to combine it with premise selection methods and ATPs. This was designed to create potential intermediate lemmas that help an ATP to find complex proofs.

While these are just first experiments, it seems that statistical concept matching can occasionally already come up with plausible conjectures without resorting to the (in large libraries rather impossible) brute-force term enumeration methods. So far we do not even use any of the manually invented heuristic methods such as those pioneered by Lenat [11] and Fajtlowicz [3], and rather rely on a data-driven approach. Such heuristics and other methods could be combined with the statistical ones.

We can likely improve the matching algorithm by allowing the concepts to be represented by more complex term structures [14]. This may help us to connect concepts from more different domains. In the same direction, we could also relax our concept of properties to allow matching with errors. A more generic solution would be to try different shapes of theorems using substitutions trees or genetic programming, but this might need efficient implementation.

We can also modify how the matching algorithm and the AI-ATP system are combined. A simple approach is to enhance the premise selection algorithm of the AI-ATP system with the discovered similarities between concepts. In our experiments we also observe an increasing number of conjectures given by the

[11] `http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/rusub_4.html#K6`
[12] `http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#K3`
[13] `http://mizar.cs.ualberta.ca/~mptp/7.13.01_4.181.1147/html/mathmorp.html#T26`

number of possible substitutions. A heuristic semantic evaluation could complement the substitutions scores to estimate the likely of a conjecture to be true.

Acknowledgments

This work has been supported by the Austrian Science Fund (FWF) grant P26201 and by the European Research Council (ERC) grant AI4REASON.

References

1. Grzegorz Bancerek, Czeslaw Bylinski, Adam Grabowski, Artur Kornilowicz, Roman Matuszewski, Adam Naumowicz, Karol Pak, and Josef Urban. Mizar: State-of-the-art and beyond. In Manfred Kerber, Jacques Carette, Cezary Kaliszyk, Florian Rabe, and Volker Sorge, editors, *Intelligent Computer Mathematics - International Conference, CICM 2015*, volume 9150 of *Lecture Notes in Computer Science*, pages 261–279. Springer, 2015.
2. Jasmin C. Blanchette, Cezary Kaliszyk, Lawrence C. Paulson, and Josef Urban. Hammering towards QED. *J. Formalized Reasoning*, 9(1):101–148, 2016.
3. Siemion Fajtlowicz. On conjectures of graffiti. *Discrete Mathematics*, 72(1-3):113–118, 1988.
4. Thibault Gauthier and Cezary Kaliszyk. Matching concepts across HOL libraries. In Stephen Watt, James Davenport, Alan Sexton, Petr Sojka, and Josef Urban, editors, *Proc. of the 7th Conference on Intelligent Computer Mathematics (CICM'14)*, volume 8543 of *LNCS*, pages 267–281. Springer Verlag, 2014.
5. Thibault Gauthier and Cezary Kaliszyk. Sharing HOL4 and HOL light proof knowledge. In Martin Davis, Ansgar Fehnker, Annabelle McIver, and Andrei Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, volume 9450 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2015.
6. Adam Grabowski, Artur Kornilowicz, and Adam Naumowicz. Mizar in a nutshell. *Journal of Formalized Reasoning*, 3(2):153–245, 2010.
7. Thomas Hales. *Dense Sphere Packings: A Blueprint for Formal Proofs*, volume 400 of *London Mathematical Society Lecture Note Series.* Cambridge University Press, 2012.
8. Cezary Kaliszyk and Josef Urban. Learning-assisted automated reasoning with Flyspeck. *Journal of Automated Reasoning*, 53(2):173–213, 2014.
9. Cezary Kaliszyk and Josef Urban. MizAR 40 for Mizar 40. *J. Autom. Reasoning*, 55(3):245–256, 2015.
10. Laura Kovács and Andrei Voronkov. First-order theorem proving and Vampire. In Natasha Sharygina and Helmut Veith, editors, *Proceedings of the 25th International Conference on Computer Aided Verification (CAV)*, volume 8044 of *LNCS*, pages 1–35. Springer, 2013.
11. Douglas Lenat. *An Artificial Intelligence Approach to Discovery in Mathematics.* PhD thesis, Stanford University, Stanford, USA, 1976.
12. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS.* Springer, 2002.

13. Florian Rabe. The MMT API: A generic MKM system. In Jacques Carette, David Aspinall, Christoph Lange, Petr Sojka, and Wolfgang Windsteiger, editors, *Proc. of the 6th Conference on Intelligent Computer Mathematics (CICM'13)*, volume 7961 of *LNCS*, pages 339–343. Springer, 2013.
14. Jiří Vyskočil, David Stanovský, and Josef Urban. Automated Proof Compression by Invention of New Definitions. In Edmund M. Clarke and Andrei Voronkov, editors, *LPAR (Dakar)*, volume 6355 of *LNCS*, pages 447–462. Springer, 2010.