Automatic Conjecturing of P-Recursions Using Lifted Inference

Jáchym Barvínek[1], Timothy van Bremen[2], Yuyi Wang[3], Filip Železný[1], and Ondřej Kuželka[1]

[1] Czech Technical University in Prague, Prague, Czech Republic
[2] KU Leuven, Leuven, Belgium
[3] ETH Zurich, Zurich, Switzerland

Abstract. Recent progress in lifted inference algorithms has made it possible to solve many non-trivial counting tasks from enumerative combinatorics in an automated fashion, by casting them as first-order model counting problems. Algorithms for this problem typically output a single number, which is the number of models of the first-order logic sentence in question on a given domain. However, in the combinatorics setting, we are more interested in obtaining a mathematical formula that holds for any given structure size. In this paper, we show that one can use lifted inference algorithms to conjecture linear recurrences with polynomial coefficients, one such class of formulas of interest.

Keywords: Lifted inference · Weighted Model Counting · Conjectures.

1 Introduction

In this paper we study the connections between enumerative combinatorics and *first order model counting* (FOMC), which is the problem of computing the number of models of a given first-order logic sentence. In enumerative combinatorics, one is typically interested in counting structures that satisfy some given properties; these structures can be graphs, sets, functions etc. Many enumerative combinatorics problems can be equivalently stated as FOMC problems. For instance, the problem of counting all labeled graphs on n vertices can be equivalently seen as counting the number of models of the sentence $(\forall x : \neg e(x, x)) \wedge (\forall x \forall y : e(x, y) \Rightarrow e(y, x))$ on the domain $\Delta = \{1, 2, \ldots, n\}$.

The main appeal of FOMC for enumerative combinatorics is the availability of a growing body of results identifying tractable classes of FOMC problems. In a seminal result, Van den Broeck [14] and Van den Broeck, Meert and Darwiche [15] proved that computing FOMC for any sentence in the two-variable fragment of first-order logic, $\mathbf{FO}^2$, can be done in time polynomial in the size of the domain. Subsequently, Beame, Van den Broeck, Gribkoff and Suciu [1], showed that, in general, this is not the case for the three-variable fragment. However, this does not mean that there are no tractable classes of FOMC problems beyond $\mathbf{FO}^2$. Two tractable fragments, called $\mathbf{S}^2\mathbf{FO}^2$ and $\mathbf{S}^2\mathbf{RU}$, were identified in [9]. Later, Kuusisto and Lutz [10] extended the tractability results for $\mathbf{FO}^2$ by

allowing the addition of a single functionality constraint, which has been recently further generalized by Kuželka [11] into the two variable fragment of first-order logic with counting quantifiers $\exists^{=k}$, $\exists^{\geq k}$ and $\exists^{\leq k}$, also known as the $\mathbf{C}^2$ fragment [8]. The latter fragment already allows expressing counting problems over non-trivial structures such as k-regular graphs.

One shortcoming of FOMC for applications in enumerative combinatorics is the fact that, when we run a FOMC algorithm on some problem, it always gives us a single number for the given domain size. However, for enumerative combinatorics, we would prefer a more analytic solution. For instance, consider the $\mathbf{C}^2$ sentence $(\forall x \exists^{=1} y : F(x,y)) \wedge (\forall y \exists^{=1} x : F(x,y))$, which asserts that the relation F is a permutation. Computing the FOMC of this sentence on domains of sizes 1, 2, 3, 4 gives us the results 1, 2, 6, 24, which we know are factorials of the domain sizes. Hence, ideally we would want to obtain a general solution of the form $n!$, or at least $a_n = n \cdot a_{n-1}$, $a_0 = 1$. The latter expression is a recurrence equation. It turns out that for many counting problems studied in combinatorics, there exist such linear recurrences with coefficients that are polynomials in n (i.e. in the domain size). In this paper we test whether one can use lifted algorithms for FOMC to conjecture such recurrent equations. We show that we are able to rediscover recurrent equations for a diverse collection of counting problems expressible in $\mathbf{C}^2$ and even conjecture new ones, such as for problems of counting the number of 2-regular-2-colored labelled graphs and 2-regular-3-colored labelled graphs for which no known recurrence exists. In particular, one of the sequences for these problems appears in the *The On-Line Encyclopedia of Integer Sequences* (OEIS[4]) but its recurrence does not and, for the other one, OEIS does not even contain the sequence. Although there have been previous works on automatic conjecture making in mathematics, e.g. works of Colton and his colleagues [5, 4] in number theory, our work is, to our best knowledge, the first that allows one to automatically generate enumerative-combinatorics conjectures about combinatorial structures specified declaratively in a fragment of first-order logic. The key component that allows this approach to work are lifted inference algorithms [14, 15, 11] without which we would not even be able to get data for generating the conjectures.

2 Background

In this section, we give some background on first-order logic, the FOMC problem and P-recursive sequences.

2.1 First-order Logic and Model Counting

We deal with the function-free, finite domain fragment of first-order logic. An *atom* of arity k takes the form $P(x_1, \ldots, x_k)$, where P/k comes from a vocabulary of *predicates* (also called *relations*), and each argument x_i is a logical variable

[4] https://oeis.org

from a vocabulary of variables. A *literal* is an atom or its negation. A *formula* is formed by connecting one or more literals together using conjunction or disjunction. A formula may optionally be surrounded by one or more quantifiers of the form $\exists x$ or $\forall x$, where x is a logical variable. A logical variable in a formula is said to be *free* if it is not bound by any quantifier. A formula with no free variables is called a *sentence*. We follow the usual semantics of first-order logic.

In this paper we restrict ourselves to the two-variable fragment of first-order logic with counting quantifiers, which is usually referred to as the $\mathbf{C}^2$ fragment [8]. This fragment is obtained by restricting the allowed sentences to contain only two variables (w.l.o.g. we can assume that these variables are x and y) and allowing quantifiers $\exists^{=k}$, $\exists^{\leq k}$, $\exists^{\geq k}$ together with the standard $\forall$ and $\exists$ quantifiers. The quantifiers $\exists^{=k}$, $\exists^{\leq k}$, $\exists^{\geq k}$ stand for *exist exactly k*, *exist at most k* and *exist at least k*, respectively.

Example 1. A function $f : \Delta \to \Delta$ is called an involution if $f(f(x)) = x$ for all $x \in \Delta$. If we want to encode involutions in $\mathbf{C}^2$, we use the sentence: $\Psi = (\forall x \exists^{=1} y : f(x, y)) \wedge (\forall x \forall y : f(x, y) \Rightarrow f(y, x))$. Here, the first conjunct uses the counting quantifier $\exists^{=1}$ to force the relation f to be a function (i.e. to have exactly one value y for every value of x) and the second conjunct forces it to be involutive.

Below, we define first-order model counting.

Definition 1 (First-order model count) *The first-order model count (FOMC) of a sentence ϕ over a domain of size n is defined as:*

$$\mathsf{FOMC}(\phi, n) = |\mathsf{models}_n(\phi)|$$

where $\mathsf{models}_n(\phi)$ denotes the set of all models of ϕ over the domain $\Delta = \{1, \ldots, n\}$. We call the sequence of numbers $a_n = \mathsf{FOMC}(\phi, n)$ the FOMC sequence of ϕ.

Example 2. Consider the sentence $\Psi = \forall x \exists^{=1} y : f(x, y) \wedge \forall y \exists^{=1} x : f(x, y)$. What is the FOMC of this sentence over the domain $\Delta = \{1, 2\}$? To answer this question, we can enumerate the models of Ψ which in this case are $\{f(1, 1), f(2, 2)\}$ and $\{f(1, 2), f(2, 1)\}$, so the answer is that FOMC is 2 in this case. In general, since we know the models of Ψ correspond to permutations, we also know the answer must be $n!$ when $|\Delta| = n$ even without enumerating all models explicitly.

Lifted inference [12–14, 7] studies ways to compute the FOMC much faster than by direct enumeration of models.[5] An important notion from the lifted inference literature is that of *domain liftability* [14], which we define below for the case of FOMC.

[5] Most algorithmic results in the lifted inference literature are presented for weighted first-order model counting (WFOMC), but for the combinatorics applications that we consider in this paper, it will be mostly sufficient to restrict our attention to FOMC even though there is, in fact, WFOMC under the hood of the algorithms that we use for FOMC—this is because existing algorithms use weights to compute FOMC with existential quantifiers [15] and with counting quantifiers [11].

Definition 2 (Domain liftability) *An algorithm for computing FOMC in a fragment of first-order logic is said to be domain-lifted if it runs in time polynomial in the size of the domain for any fixed sentence from this fragment (the polynomial may depend on the sentence). A fragment of first-order logic is said to be domain-liftable if such a domain-lifted algorithm exists for it.*

In this paper we rely on the following result asserting the tractability of computing the FOMC for sentences from the $\mathbf{C}^2$ fragment of first-order logic [11], which builds on previous works of Van den Broeck [14], Van den Broeck, Meert and Darwiche [15] and Kuusisto and Lutz [10].

Theorem 1 (Kuželka, 2021 [11]). *The fragment of first-order logic limited to two variables with counting quantifiers, $\mathbf{C}^2$, is domain-liftable.*

2.2 P-Recursive Sequences

In this paper we are interested in finding *P-recursive relations* for FOMC sequences. First, we briefly introduce P-recursivity.

Definition 3 *A sequence of integers $\{a_n\}_{n=0}^{\infty}$ is called* P-recursive *if there exists $k \in \mathbb{N}$ and polynomials p_i for each $i = 0, ..., k$ with integer-valued coefficients such that for each $n > k$ it holds:*

$$\sum_{i=0}^{k} p_i(n) a_{n-i} = 0. \tag{1}$$

Here, we call the number k to be the order *of the sequence and maximum degree*[6] *$d \geq 0$ of the polynomials p_i to be the* degree *of the sequence. Furthermore, we denote with $c_{i,j}$ the coefficient of p_i at the $j-th$ power. This way, we can rewrite (1) in more detail as:*

$$\sum_{i=0}^{k} \sum_{j=0}^{d} a_{n-i} c_{i,j} n^j = 0. \tag{2}$$

Example 3. The Fibonacci sequence is P-recursive with $d = 0, k = 2, p_0(n) = -1, p_1(n) = p_2(n) = 1$.

Example 4. The sequence of factorials $n!$ is P-recursive with $d = 1, k = 1, p_0(n) = -1, p_1(n) = n$.

P-recursive sequences are of interest because the recurrence relations offer a straightforward and computationally inexpensive way to represent and evaluate the sequence terms, and perform other computations. See, for example, [6] for more details.

We will call the closed form expression of the form $a_{n+1} = f(a_n)$ derivable from (1) for some P-recursive sequence $\{a_n\}_{n=0}^{\infty}$ the *P-recursive relation* corresponding to the sequence. In this paper we will be trying to conjecture such relations automatically.

[6] It is convenient to consider the zero polynomial to have degree 0 and a single coefficient 0.

3 Approach

In this paper we take a pragmatic approach for conjecturing P-recursive relations. We assume that we are given a sentence in $\mathbf{C}^2$, encoding the combinatorial structures that we want to count. For instance, to count functions from Δ to Δ, the $\mathbf{C}^2$ sentence would be $\forall x \exists^{=1} y : F(x, y)$. We use an FOMC algorithm[7] to generate a sequence of numbers. We generate as many terms of this sequence as computationally tractable or up to 50 when the recurrence is already known and more is not needed. We then input the computed sequence terms into the method described below, which itself determines the number of samples from the sequence required to learn a conjecture. The conjecture, if found, is validated against the remaining terms not sampled to conjecture the equation. Success of the method does not constitute a proof that the equations we find are correct, or even that the sequence is P-recursive, but it allows us to be reasonably confident about them as conjectures. Of course, our confidence in them depends on the length of the validation sequence. Failure of the method to find a conjecture indicates that the sequence is either not P-recursive, or it is P-recursive with such a high order and/or degree that the input sequence provides insufficient information to reconstruct the recurrence equation.

3.1 Conjecturing Recurrence Relations

Our method resembles the one described in [2], but is simpler and adapted to our specific use case. Suppose we have the first l terms of a sequence $(a_0, ..., a_{l-1})$, and we are trying to conjecture a P-recursive relation for given values of the metaparameters k, d. From Equation (2), we can directly obtain a system of $l - k$ linear equations with unknowns $c_{i,j}$. The right hand side of each equation is zero, so we can view the problem as looking for the kernel of a certain matrix M depending on $k, d, \{a_n\}_{n=0}^{l-1}$ and implicitly defined by Equation (2). Specifically, the system of equations obtained is equivalent to the matrix equation $\mathbf{M} \cdot \mathbf{c} = \mathbf{0}$ with:

$$\mathbf{M}_{i,j} = a_{i+k-(j \mod (k+1))}(i+k)^{\left\lfloor \frac{j}{k+1} \right\rfloor} \tag{3}$$

$$\mathbf{c}_j = c_{\left\lfloor \frac{j}{k+1} \right\rfloor, j \mod (k+1)} \tag{4}$$

for $i = 0, \ldots, l-1-k$ and $j = 0, \ldots, (k+1)(d+1)-1$. (This is mostly reindexing resulting from flattening the table $c_{i,j}$ from (2) into the vector $\mathbf{c}$. See how the quotients $\left\lfloor \frac{j}{k+1} \right\rfloor$ correspond to i's in (2) and the remainders $j \mod (k+1)$ correspond to j's, and the $i + k$ here corresponds to n in (2).)

Note, that for a P-recursive sequence defined by polynomials p_i, we can generate a linear space of equivalent representations: certainly, we can multiply each of the polynomials p_i in (1) by another fixed nonzero polynomial to obtain a

[7] The implementation that we use is based on the algorithm described in [11]. This algorithm needs access to an $\mathbf{FO}^2$ weighted FOMC oracle, for which we use our own optimized version [3] of the algorithm described in [1, Appendix C].

different representation of the same sequence. In general sometimes even different linearly independent vectors of $c_{i,j}$ can describe equivalent representations.

Example 5. Consider the sequence $a_n = n^4$. It can be verified that for $k = d = 2$ any triple of polynomials $(p_0, p_1, p_2) \in \text{span}\{(-5n^2+14n-10, -32n-32, 5n^2-6n+2), (-40n^2+101n-65, 40n^2+48n+112, 11n-7)\}$ gives a valid P-recursive formula for this sequence. Our method unambiguously identifies this linear space when provided $\{n^4\}_{n=0}^{8}$.

Now, we consider the nullspace of the matrix $\mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l-1})$, which we denote as $\ker \mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l-1})$, to describe a P-recursivity conjecture for a sequence starting with $\{a_n\}_{n=0}^{l-1}$ if the following property holds: There exists a number $l^* < l$, such that

$$\ker \mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l^*-2}) \neq \ker \mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l^*-1}) \tag{5}$$

but for each $l', l^* < l' \leq l$ it holds that:

$$\ker \mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l'-1}) = \ker \mathbf{M}^{k,d}(\{a_n\}_{n=0}^{l^*-1}) \neq \{\mathbf{0}\}. \tag{6}$$

This could equivalently be rephrased in machine learning terminology as follows: We consider the kernel to be a conjecture if it correctly predicts all the following sequence terms after l^* but is not unambiguously learnable from less than l^* samples. The difference $l - l^*$ is a measure of the strength of corroborating evidence for the conjecture following from the extra sequence terms available and can be seen as an analogue of the size of validation set in machine learning. Note that to construct the polynomials p_i from the kernel, we can take any linear combination of its basis vectors. In practice, the dimension of this space is often 1.

To find the metaparameters k, d we used a simple grid-search iterating over the pairs $(k, d) \in \{0, \ldots l-1\}^2$ in the order of increasing $k + d$. If no conjecture can be found this way, the algorithm exits with a failure status.

Example 6. Suppose we are given the five term sequence $\{a_n\}_{n=0}^{4} = (1, 1, 2, 6, 24)$ and $k = d = 1$. The corresponding matrix is:

$$\mathbf{M} = \begin{pmatrix} a_2 & a_1 & 1a_2 & 1a_1 \\ a_3 & a_2 & 2a_3 & 2a_2 \\ a_4 & a_3 & 3a_4 & 3a_3 \\ a_5 & a_4 & 4a_5 & 4a_4 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & 4 & 2 \\ 6 & 2 & 18 & 6 \\ 24 & 6 & 96 & 24 \end{pmatrix} \tag{7}$$

The kernel has dimension one and is $\ker \mathbf{M} = \text{span}\{(-1, 0, 0, 1)\}$. This would be the same if we dropped the last term of the sequence, but different if we dropped the last two. This is equivalent to dropping the bottom row(s) of $\mathbf{M}$. Therefore $l^* = 4$. We can use this basis vector to write a recurrence relation:

$$(-1 \cdot n^0 + 0 \cdot n^1)a_n + (0 \cdot n^0 + 1 \cdot n)a_{n-1} = 0 \tag{8}$$

which makes the conjectured sequence $a_n = na_{n-1} = n!$, whose correctness can be validated by the sample a_4 not needed to obtain this result.

Property P-recursive relation conjectured	OEIS ID	k	d	Known	l^*	l	Time a_0, a_1, a_2, a_3
Permutations $a_n = na_{n-1}$	A000142	1	1	Yes	4	50	1h 3min 1, 1, 2, 6
Derangements $a_n = (n-1)(a_{n-2} + a_{n-1})$	A000166	2	1	Yes	7	50	12min 1, 0, 1, 2
Involutions $a_n = a_{n-1} + (n-1)a_{n-2}$	A000085	2	1	Yes	7	50	4min 1, 1, 2, 4
1-regular graphs (involutive derangements) $a_n = (2n-1)a_{n-1}$. Here we use $n = 2\|\Delta\|$ as the property is trivially unsatisfiable for odd $\|\Delta\|$.	A001147	1	1	Yes	4	50	1h 5min 1, 1, 3, 15
2-regular graphs $2a_n = (n-1)(2a_{n-1} + (n-2)a_{n-3})$	A001205	3	2	Yes	14	50	6min 1, 0, 0, 1
2-regular $\cap$ 2-colored graphs $a_n = 2(n-1)(2n-1)((2n-3)a_{n-2} + a_{n-1})$. Also using $n = 2\|\Delta\|$.	A054479	2	2	**No**	15	32	17h 50min 1, 0, 6, 120
2-regular $\cap$ 3-colored graphs $a_n = (n-1)(a_{n-1} + (n-2)(2a_{n-2} + 3a_{n-3} + 6(n-3)a_{n-4}))$	N/A	4	3	**No**	23	31	11h 40min 1, 0, 0, 6
2-colored graphs Does not appear to be P-recursive.	A047863	–	–	–	–	400	9min 1, 2, 6, 26
3-colored graphs Does not appear to be P-recursive.	A191371	–	–	–	–	400	20min 1, 3, 15, 123

Table 1. Examples of applying our conjecturing method to some FOMC sequences of properties in $\mathbf{C}^2$. The "Known" column indicates whether a P-recurrence formula could be found in OEIS or other sources. For the sequences marked as "No", we consider the conjectured recurrence relation to be a novel discovery (to the best of our knowledge). The sequences marked as "Yes" were correctly rediscovered by our automated method. The l^* column is the minimum number of samples required to learn the sequence in the sense defined in Section 3. The l column is the actual number of sequence terms we were able to compute within the total time shown in the last column.

4 Experiments

We computed the FOMC sequences for several logical properties and attempted to compute as many terms as possible in reasonable time. We then used those sequences to find a conjecture about P-recursivity. For some of the properties, a P-recursive relation is already known and our algorithm merely reproduces it. However, for two of those properties, the algorithm conjectured a relation for which we found no such relation on OEIS. The FOMC sequences were computed using our own implementation as explained in Section 3, and ran on a single 3.7GHz Intel i5-9600KF processor core with up to 32 GiB memory available. This was computationally the most difficult part.

The code for finding the recurrence relation was implemented in Mathematica 12 [16]. This process was usually quick and found a solution within seconds if it existed with low degree and order. For sequences which do not seem to be P-recursive, this is comparably slower as the grid search is attempting many high-valued metaparameter candidates. The results are summarised in Table 1.

Since we were using the Mathematica software, we noticed that some of the recurrences could automatically be converted to closed-forms. For example, for the 1-regularity, we obtained the formula[8] $a_n = 2^n \pi^{-\frac{1}{2}} \Gamma(\frac{1}{2} + n)$. We got similar symbolic solutions also for permutations and derangements. For the rest of the problems, Mathematica was not able to find a solution of the recurrences in terms of standard special functions. Methods for solving recurrence equations could thus be used to extend our pipeline with closed forms at output where available.

5 Conclusions

In this short paper we proposed a methodology for generating conjectures about P-recursivity of combinatorial sequences using techniques from lifted inference. We demonstrated the potential of the approach by showing that we can rediscover non-trivial recurrence relations from the literature, as well as conjecture new ones. It is likely that these conjectures could be proven by an expert enumerative combinatorialist. In the future, we want to move from conjecturing the recurrences to proving them algorithmically, thus, if we exaggerate a bit, making an *automatic enumerative combinatorialist*.

Acknowledgements

JB and OK were supported by the Czech Science Foundation project "Generative Relational Models" (20-19104Y). JB was also supported by a donation from X-Order Lab. TvB was supported by the Research Foundation – Flanders (G095917N). FZ was supported by the Czech Science Foundation project 20-29260S.

[8] Here, the expression obtained from Mathematica was originally expressed using the Pochhammer symbol which we rewrote using the gamma function.

References

1. Beame, P., Van den Broeck, G., Gribkoff, E., Suciu, D.: Symmetric weighted first-order model counting. In: PODS. pp. 313–328. ACM (2015)
2. Berthomieu, J., Faugère, J.C.: Guessing linear recurrence relations of sequence tuples and p-recursive sequences with linear algebra. In: Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation. pp. 95–102 (2016)
3. van Bremen, T., Kuzelka, O.: Faster lifting for two-variable logic using cell graphs. In: Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence, UAI 2021 (2021)
4. Colton, S.: Automated conjecture making in number theory using hr, otter and maple. J. Symb. Comput. **39**(5), 593–615 (2005). https://doi.org/10.1016/j.jsc.2004.12.003, https://doi.org/10.1016/j.jsc.2004.12.003
5. Colton, S., Bundy, A., Walsh, T.: Automatic invention of integer sequences. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA. pp. 558–563 (2000)
6. Flajolet, P., Sedgewick, R.: Analytic combinatorics. cambridge University press (2009)
7. Gogate, V., Domingos, P.M.: Probabilistic theorem proving. In: UAI. pp. 256–265. AUAI Press (2011)
8. Grädel, E., Kolaitis, P.G., Vardi, M.Y.: On the decision problem for two-variable first-order logic. Bull. Symb. Log. **3**(1), 53–69 (1997)
9. Kazemi, S.M., Kimmig, A., Van den Broeck, G., Poole, D.: New liftable classes for first-order probabilistic inference. In: NIPS. pp. 3117–3125 (2016)
10. Kuusisto, A., Lutz, C.: Weighted model counting beyond two-variable logic. In: LICS. pp. 619–628. ACM (2018)
11. Kuzelka, O.: Weighted first-order model counting in the two-variable fragment with counting quantifiers. J. Artif. Intell. Res. **70**, 1281–1307 (2021)
12. Poole, D.: First-order probabilistic inference. In: IJCAI. pp. 985–991. Morgan Kaufmann (2003)
13. de Salvo Braz, R., Amir, E., Roth, D.: Lifted first-order probabilistic inference. In: IJCAI. pp. 1319–1325. Professional Book Center (2005)
14. Van den Broeck, G.: On the completeness of first-order knowledge compilation for lifted probabilistic inference. In: NIPS. pp. 1386–1394 (2011)
15. Van den Broeck, G., Meert, W., Darwiche, A.: Skolemization for weighted first-order model counting. In: KR. AAAI Press (2014)
16. Wolfram Research, Inc.: Mathematica, Version 12.2 (2020), https://www.wolfram.com/mathematica, champaign, IL, 2020