Lifted Reasoning for Combinatorial Counting
ID 4398

Abstract

Combinatorics math problems are often used as a benchmark to test human cognitive and logical problem-solving skills. These problems are concerned with counting the number of solutions that exist in a specific scenario that is often sketched in natural language. Humans are adept at solving such problems as they can identify commonly occurring structures in the questions for which a closed-form formula exists for computing the answer. These formulas exploit the exchangeability of objects and symmetries to avoid a brute-force enumeration of all possible solutions. Unfortunately, current AI approaches are still unable to solve combinatorial problems in this way. This paper aims to fill this gap by developing novel AI techniques for representing and solving such problems. It makes the following three contributions. First, we propose a declarative language that is able to represent a wide range of combinatorial reasoning problems. Second, we propose a novel lifted solving algorithm bridging probabilistic inference techniques and constraint programming. Third, we show empirically how our solver efficiently solves a wide range of combinatorics math problems.

1 Introduction

Some of the most interesting challenges in AI come from tasks that humans are able to solve elegantly by means of abstract reasoning. One of the fundamental goals of AI is to outperform humans on tests associated to intelligence and advanced cognitive skills, like puzzles [Chesani *et al.*, 2017], games [Silver *et al.*, 2016] and math and probability problems [Mitra and Baral, 2016; Roy and Roth, 2018; Dries *et al.*, 2017]. For this reason, an important source of inspiration to develop better AI tools comes from quizzes, exercises and exams that test cognitive skills, such as SAT, GMAT or GRE. These tests require logical and mathematical proficiency along with text comprehension skills. The class of problems we adopt in this paper is combinatorics math problems. The task is to count the number of possible configurations of a setting, described in natural language. Combinatorics math problem occupy an interesting position between *counting constraint satisfaction problems* (#CSP) and probability. On the one hand, they can, as we will show, be modeled as instances of #CSPs and we will develop a declarative language for this. On the other hand, solving combinatorial problems requires reasoning about high-level symmetries, in a way that resembles *lifted* probabilistic inference [Poole, 2003] for probabilistic logics. This raises the question as to whether the lifted reasoning techniques for probabilistic inference can be adapted for combinatorics problems, and given that we represent combinatorics problems as counting constraint satisfaction problems, whether lifted inference can be adapted to these problems as well. This question is answered positively in this paper by combining the strengths of constraint languages and lifted inference. In particular, we present the following contributions:

- We define a declarative language, CoLa, to express combinatorial problems (Section 2).
- We define concepts and algorithms that allow for an efficient (lifted) solving strategy for counting constraint satisfaction problems (Section 4)
- We implement a solver, CoSo, for a class of combinatorics problems (Section 5) showing an application of our approach.
- We compare CoLa and CoSo to alternative frameworks for modelling counting problems (Sections 3 and 6).

2 Combinatorics math problems

In a combinatorics math problem, the task is to determine how many configurations of a finite set of entities satisfy a given set of constraints. In the most basic form, the constraints simply define the type (and often size) of configuration (subset, sequence, partition, etc), and all we know about the entities is whether or not they are distinguishable, i.e., whether we can tell them apart or not. Consider the following basic problems:

P1 In how many different ways can Ann, Bob, Carl and Dan queue at the bus stop?

P2 In how many ways can we distribute fifty nails into five identical boxes?

P1 asks for different permutations (queues) of four distinguishable entities (people), P2 asks for integer partitions

$f : X \to Y$	f_{any}	f_{inj}	f_{sur}
$\neq$	sequences $Y = \ldots; a \in [\|\|Y]; \#a = \|X\|;$	permutations $Y = \ldots; a \in [\|Y]; \#a = \|X\|$	compositions $X = \ldots; a \in C(X); \#a = \|Y\|;$
$=^X$	multisubsets $Y = \ldots; a \in \{\|\|Y\}; \#a = \|X\|;$	subsets $Y = \ldots; a \in \{\|Y\}; \#a = \|X\|;$	integer compositions $X^= = \ldots; a \in C(X); \#a = \|Y\|;$
$=_Y$	partitions $X = \ldots; a \in P(X); \#a \leqslant \|Y\|;$	partitions $X = \ldots; a \in P(X); \#a = \|X\|;$	partitions $X = \ldots; a \in P(X); \#a = \|Y\|;$
$=_Y^X$	integer partitions $X^= = \ldots; a \in P(X); \#a \leqslant \|Y\|;$	integer partitions $X^= = \ldots; a \in P(X); \#a = \|X\|;$	integer partitions $X^= = \ldots; a \in P(X); \#a = \|Y\|;$

Table 1: Twelvefold Way: basic combinatorial configurations and corresponding CoLa statements; cf. **Configurations** in Section 2 for details.

(with up to five summands) of fifty indistinguishable entities (nails). In general, however, these problems include additional properties of entities, and additional constraints based on these properties, as in the following problem:

Example 1. *(from the dataset of [Dries* et al.*, 2017])*

P3 *A shipment of 12 TVs contains 3 defective ones. In how many ways can a hotel purchase 5 of these TVs and receive at least 2 of the defective ones?*

Here, the set of entities are the 12 TVs, which fall into two subsets based on whether they are defective or not. All TVs are distinguishable, the basic configuration is a subset of size 5 (the purchase), and the additional constraint on the configuration is that it contains at least two of the defective TVs.

In the remainder of this section, we formalize the class of problems considered, and introduce CoLa, our modeling language for these problems.

Entities and Sets In CoLa, *entities* are represented by constant symbols, i.e., $e_1, e_2, \ldots, e_n$. The *universe* is the set of all entities. A set S can be defined by enumeration, $S = \{e_1, e_2, e_3\}$. A *set formula* is a set, or a combination of sets using the set-operators $\cap, \cup, \overline{\ }$, where $\overline{F}$ is the set-complement w.r.t. the universe. The entities of a set S can be declared *indistinguishable* with the symbol $^=$, $S^= = \{e_1, e_2, e_3\}$.

Configurations To characterize the basic configurations, we follow the *Twelvefold way* [Stanley, 2012], which describes configurations through functions $f : X \to Y$ between two finite sets X, Y. As illustrated in Table 1, two dimensions are used to classify problems. The first is whether the function f is arbitrary (f_{any}), or constrained to be either injective (f_{inj}) or surjective (f_{sur}). The second is concerned with the (in)distinguishability of entities, where we have all elements distinguishable ($\neq$), elements in X indistinguishable and those in Y distinguishable ($=^X$), elements in X distinguishable and those in Y indistinguishable ($=_Y$), or elements in both X and Y indistinguishable ($=_Y^X$).

As indicated in the table, the resulting classes are well-known in combinatorics. CoLa uses the following notation for the different configurations, where S is a set of entities:

- $\{|S\}$ denotes the set of all *subsets* of S;
- $\{||S\}$ denotes the set of all *multisubsets* of S: elements of S can occur multiple times;
- $[|S]$ denotes the set of *permutations* $[e_1, ..., e_n]$ of length $n \in \{1, \ldots, |S|\}$ formed by different elements $e_i \in S$;
- $[||S]$ denotes the set of *sequences* $[e_1, ..., e_n]$ of length $n \in \{1, \ldots, |S|\}$ formed by elements $e_i \in S$;
- $P(S)$ denotes the set of all *partitions* of S;
- $C(S)$ denotes the set of all *compositions* of S where a composition is a sequence of sets $[S_1, ..., S_n]$ such that $S = \cup_i S_i$ and $S_i \cap S_j = \varnothing$ for all $i, j \in \{1 \ldots n\}, i \neq j$.

For such a set A of configurations, the statement $a \in A$ introduces a new set a that is constrained to be a member of A. For any set formula F, $\#F$ denotes the size of F, i.e., the number of its elements. *Size constraints* are of the form $\#F \star n$ with $\star \in \{<, \leqslant, =, >, \geqslant, \neq\}$ and n a natural number. The CoLa encoding of a configuration introduces a new set of the corresponding type together with a size constraint as listed in Table 1.

Example 2. *The following CoLa statements encode P1:*
$people = \{ann, bob, carl, dan\}; \quad queue \in [|people]; \quad \#queue = 4;$
P2 is represented by the following CoLa statements:

$$nails^= = \{n_1, \ldots, n_{50}\}; boxes \in P(nails); \#boxes \leqslant 5$$

Properties and Additional Constraints CoLa identifies a property with the set of entities that have the property, and treats them as any other set, i.e., properties can appear in set formulas, and in size constraints over these set formulas. If a set S is a permutation, sequence or composition, the value of the ith element of S can be constrained using a *positional constraint* of the form $S[i] = x$, where x is of the same form as the elements of S (e.g., $queue[2] = dan$), or $S[i] \in X$, where the elements of S and X are of the same form (e.g., $queue[2] \in boys$ with $boys = \{bob, carl, dan\}$).

Example 3. ***P3** in Example 1 is encoded as follows:*
$tvs = \{s_1, \ldots, s_{12}\}; \quad defective = \{s_1, s_2, s_3\};$
$purchased \in \{|\, tvs\}; \quad \#purchased = 5;$
$\#(defective \cap purchased) \geqslant 2;$

3 Model Counting

We model combinatorics math problems as *counting constraint satisfaction problems* (#CSPs). A *constraint satisfaction problem* (CSP) is a tuple $\langle \mathcal{V}, D, C \rangle$ where $\mathcal{V} = \{V_1, \ldots, V_n\}$ is a finite set of variables, $D = \{D_1, \ldots D_n\}$ is a set of domains, i.e., D_i is a finite set of values that V_i can take, and we write $D_\times = D_1 \times \cdots \times D_n$. C is a set of constraints. A *constraint* is a pair $(\mathbf{v}, R)$ where $\mathbf{v}$ is an n-tuple of $\mathcal{V}$ and $R \subseteq D_\times$ is a relation of arity n over $D_\times$. When

the arity of a constraint c is equal to the number of variables we call c a *global constraint*. An *assignment* is a function $f : \mathcal{V}^i \to D^i_\times : f((V_1, \ldots, V_i)) = (d_1, \ldots, d_i)$. f is a *partial* assignment when $i < |\mathcal{V}|$. An assignment f *satisfies* a constraint $c = (\mathbf{v}, R)$ when $f(\mathbf{v}) \in R$:$f(\mathbf{v}) \vdash c$. The task in CSPs is to find a satisfying assignment (also called a solution or a model) which is an assignment that satisfies all the constraints in C. We refer to the set of all models as $M(\mathcal{V}, D, C)$. The #CSP $MC(\mathcal{V}, D, C)$ (model counting problem) for a CSP $\langle \mathcal{V}, D, C \rangle$ then consists of finding $|M(\mathcal{V}, D, C)|$. Combinatorics problems can be expressed as #CSPs: the finite set of entities corresponds to the domain D. The possible configurations of entities are represented by the set of variables $\mathcal{V}$, the set of constraints C defines properties and valid configurations. The challenge is to efficiently compute the model count as this problem reduces to the #SAT counting problem, which is #P-complete [Valiant, 1979].

One approach would be to consider probabilistic inference which also reduces to #SAT. *Lifted* probabilistic inference techniques [Poole, 2003; Niepert and Van den Broeck, 2014] tackle this complexity by exploiting symmetries in a model to reason about groups of objects with the same characteristics. Lifted variants exist of probabilistic inference algorithms such as variable elimination [de Salvo Braz *et al.*, 2005; Taghipour *et al.*, 2012] and knowledge compilation [Van den Broeck *et al.*, 2011]. Lifting can lead to polynomial complexity results for some model classes [Van den Broeck *et al.*, 2014; Kazemi *et al.*, 2016]. Algebraic Model Counting (AMC), implemented in aProbLog, [Kimmig *et al.*, 2017] generalizes probabilistic inference, #SAT, and (weighted) model counting and shows how (first-order) knowledge compilation solves an AMC task.

Unfortunately, lifted probabilistic inference techniques are not suited for the problem at hand because they are designed for different kinds of models, typically weighted first order logic formulas, which differ from #CSPs. First, sets (and properties) would have to be modeled by unary predicates, e.g. $set(e_1), \ldots, set(e_n)$. Second, constants in such sets are always considered distinguishable, and it is unclear how this could be changed. Third, support for constraints is typically very limited. For example, in FOWMC [Van den Broeck *et al.*, 2011] the compilation rules can lift only binary constraints of the form $v \in D$ and $t_1 = t_2$, where v is a variable, D is a set of constants and t_i is either a variable or a constant. In contrast to these approaches, we will exploit high-level symmetries of global constraints (injectivity, size constraints) to perform lifted reasoning. While GC-FOVE uses the more expressive *constraint trees* to represent the set of admissible assignments to variables, it may struggle with global constraints. For example, the constraint tree for *injectivity* would correspond to an enumeration of all possible permutations of the variables, thus precluding lifted optimizations.

Another approach is to consider CSP solvers and languages such as MiniZinc [Nethercote *et al.*, 2007]. While these are a more natural choice to encode combinatorial problems, they have several shortcomings for our task. They lack explicit support for indistinguishability, that is, multisubsets and sets of sets (partitions) cannot be directly declared. More crucially, they employ a non-lifted solving technique: constraint solvers are designed to find one solution at a time. In Section 6 we show how enumerating all solutions of combinatorial problems is infeasible even for relatively small problems.

	A	B	C	Solves	Lifted
aProbLog	x	x	binary	x	x
Forclift	x	x	binary	x	✓
GC-FOVE	x	x	global	x	✓
CSPs	✓	∼	global	✓	x
Our	✓	✓	global	✓	✓

Table 2: Frameworks comparison: A=Sets, B=Configurations, C=Constraints. ∼ indicates incomplete support.

We present an application of lifted reasoning concepts to #CSPs in Sections 4 and 5. Table 2 summarizes whether the fundamental elements of combinatorics math problems can be encoded adequately in model counting frameworks, whether it is possible to solve them, and whether the solving procedure adopts lifted reasoning techniques.

4 Lifted Reasoning Over #CSPs

We now discuss how the main probabilistic lifted inference techniques can be defined in the context of #CSPs. Following the notions about lifted probabilistic inference presented in [Niepert and Van den Broeck, 2014] we say that *domain-lifted* model counting algorithms run in time polynomial in the size of the domain.

Exchangeability is a fundamental concept for lifted reasoning, since it allows reasoning over groups rather than individuals: recognizing and exploiting exchangeability brings exponential improvements compared to reasoning individually over each exchangeable assignment.

Definition 1. *A tuple of variables $\mathcal{V} = (V_1, \ldots, V_n)$ is* exchangeable *($\equiv^{\mathcal{V}}$) w.r.t. a* CSP *$\langle \mathcal{V}, D, C \rangle$ if for all satisfying assignments $(V_1 = d_1, \ldots, V_n = d_n)$ and all permutations π of $\{1, \ldots, n\}$, $(V_1 = d_{\pi(1)}, \ldots, V_n = d_{\pi(n)})$ is a satisfying assignment as well.*

Property 1. *Given a* CSP *$\langle \mathcal{V}, D, C \rangle$ and satisfying assignment $(V_1 = d_1, \ldots, V_n = d_n)$, if $(V_1, \ldots, V_n)$ is exchangeable then there are other $n! - 1$ satisfying assignments.*

Note that while the lifted inference literature often uses *exchangeable* and *indistinguishable* for the same concept [Taghipour *et al.*, 2012; Milch *et al.*, 2008], in this paper, *exchangeable* refers to a set of *variables* in a CSP as formally defined above, whereas we use *indistinguishable* as in the Twelvefold way to refer to a set of entities that cannot be told apart, i.e., *domain elements* in a CSP. Both concepts are also related to *symmetries* in constraint modeling, e.g., the vertical, horizontal and rotational symmetries of the chessboard in the famous n-queens problem.

We now define two fundamental operations of lifted inference in the context of #CSPs: splitting and shattering.

Definition 2. *Given two tuples of disjoint variables $\boldsymbol{v}_1, \boldsymbol{v}_2$ and two assignments $f_1(\boldsymbol{v}_1) = (d_1, \ldots, d_m)$, $f_2(\boldsymbol{v}_2) = d_{m+1}, \ldots, d_n)$, the* union *of the two assignments $f_1 + f_2$ is the tuple: $(d_1, \ldots, d_m, d_{m+1}, \ldots, d_n)$. We generalize*

to the union of two sets M_1, M_2 of partial assignments as $M_1 + M_2 = \bigcup_{\boldsymbol{d}_1 \in M_1} \bigcup_{\boldsymbol{d}_2 \in M_2} \{\boldsymbol{d}_1 + \boldsymbol{d}_2\}$.

Property 2. *Given two* CSP*s $\langle \mathcal{V}_1, D_1, C_1 \rangle$, $\langle \mathcal{V}_2, D_2, C_2 \rangle$, $\mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$, $MC(\mathcal{V}_1 \cup \mathcal{V}_2, D_1 \cup D_2, C_1 \cup C_2) = MC(\mathcal{V}_1, D_1, C_1) \cdot MC(\mathcal{V}_2, D_2, C_2)$ (multiplication rule).*

Definition 3. *Given a* #CSP $MC(\mathcal{V}_1 \cup \mathcal{V}_2, D, C), \mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$, *a* split *is pair of* #CSP*s $(\mathbb{S}_1, \mathbb{S}_2), \mathbb{S}_i = MC(\mathcal{V}_i, D_i, C_i)$, such that $\mathbb{S}_1 \cdot \mathbb{S}_2 = MC(\mathcal{V}, D, C)$.*

Example 4. *A* #CSP $MC(\mathcal{V}, D, C)$ *,* $V = \{V_1, V_2, V_3\}$, $D_1 = D_2 = D_3 = \{1, 2, 3\}$, $C = \{V_1 < 3, V_2 \neq V_3\}$, *can be split into* $(\mathbb{S}_1 = \langle \{V_1\}, \{D_1\}, \{V_1 < 3\} \rangle, \mathbb{S}_2 = \langle \{V_2, V_3\}, \{D_2, D_3\}, \{V_2 \neq V_3\} \rangle$*:* $MC(\mathcal{V}, D, C) = \mathbb{S}_1 \cdot \mathbb{S}_2 = 2 \cdot 6 = 12$*. Note that V_2 and V_3 are exchangeable.*

Shattering generalizes a split: as in probabilistic lifted inference it can be defined as the repeated application of the splitting operation.

Definition 4. *Given a* #CSP $MC(\mathcal{V}, D, C)$ *a* shattering*, is a set $\{\mathbb{S}_1, \ldots, \mathbb{S}_n\}$, $\mathbb{S}_i = MC(\mathcal{V}_i, D_i, C_i)$, such that $MC(\mathcal{V}, D, C) = \prod_{i=1}^{n} \mathbb{S}_i$.*

The multiplication rule holds only if the subproblems $\mathbb{S}_i$ are independent, i.e. a satisfying assignment for variables $\mathcal{V}_i$ on $\mathbb{S}_i$ is a partial assignment for the initial problem $MC(\mathcal{V}, D, C)$ independent from the others. That is, C does not contain any non-trivial constraint relating the value of a variable V_1 to the value of a variable V_2 with $V_1 \in \mathbb{S}_i, V_2 \in \mathbb{S}_j$ and $i \neq j$. Nonetheless, the solution of the #CSP can still be expressed as the combination of a set of splits of the problem, by splitting and shattering such constraints along with the problem.

Definition 5. *Given a* CSP $\langle \mathcal{V}_1 \cup \mathcal{V}_2, D_1 \cup D_2, C \rangle, \mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$, *a* constraint split *is a pair (C_1, C_2) such that for each satisfying assignment f_1, f_2 for, respectively, $\langle \mathcal{V}_1, D_1, C_1 \rangle$ and $\langle \mathcal{V}_2, D_2, C_2 \rangle$:*

$$f_1(\mathcal{V}_1) \vdash C_1 \wedge f_2(\mathcal{V}_2) \vdash C_2 \implies f_1(\mathcal{V}_1) + f_2(\mathcal{V}_2) \vdash C$$

Example 5. *Consider Example 4: $\mathbb{S}_2$ cannot be further split since the constraint $V_2 \neq V_3$ relates the choices for the two variables. A constraint split for $\{V_2 \neq V_3\}$ is $(\{V_2 = 1\}, \{V_3 \neq 1\})$: any union of satisfying assignments for the subproblems $\langle \{V_2\}, \{D_2\}, \{V_2 = 1\} \rangle$ and $\langle \{V_3\}, \{D_3\}, \{V_3 \neq 1\} \rangle$ satisfies $\mathbb{S}_2$ as well.*

While constraint splits preserve satisfiability, they do not preserve the model count, as Example 5 shows: the set of satisfying assignments for the split is a subset of the set of solutions for $\mathbb{S}_2$. A *constraint shattering* of $\langle \mathcal{V}, D, C \rangle$ is a set of constraint splits such that each split corresponds to a partition of the set of satisfying assignments for the original problem.

Definition 6. *Two constraint splits (C_1^1, C_2^1) and (C_1^2, C_2^2) for a problem $\langle \mathcal{V}_1 \cup \mathcal{V}_2, D, C \rangle, \mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$,* overlap *if there exist two satisfying assignment f_1 and f_2 such that $f_1(\mathcal{V}_1) \vdash C_1^1, f_1(\mathcal{V}_2) \vdash C_2^1, f_2(\mathcal{V}_1) \vdash C_1^2, f_2(\mathcal{V}_2) \vdash C_2^2$ and $f_1(\mathcal{V}_1) + f_1(\mathcal{V}_2) = f_2(\mathcal{V}_1) + f_2(\mathcal{V}_2)$.*

Definition 7. *Given a* CSP $\langle \mathcal{V}_1 \cup \mathcal{V}_2, D_1 \cup D_2, C \rangle, \mathcal{V}_1 \cap \mathcal{V}_2 = \varnothing$, *a* constraint shattering $\mathbb{C}$ *is a set of non-overlapping constraint splits (C_1, C_2) such that the union of the satisfying assignments of the splits is equal to $M(\mathcal{V}_1 \cup \mathcal{V}_2, D_1 \cup D_2, C)$:*

$$M(\mathcal{V}_1 \cup \mathcal{V}_2, D, C) = \bigcup_{(C_1, C_2) \in \mathbb{C}} M(\mathcal{V}_1, D_1, C_1) + M(\mathcal{V}_2, D_2, C_2)$$

Property 3. *Given a* #CSP $MC(\mathcal{V}_1 \cup \mathcal{V}_2, D_1 \cup D_2, C)$ *and a constraint shattering $\mathbb{C}$:*

$$MC(\mathcal{V}_1 \cup \mathcal{V}_2, D, C) = \sum_{(C_1, C_2) \in \mathbb{C}} MC(\mathcal{V}_1, D_1, C_1) \cdot MC(\mathcal{V}_2, D_2, C_2)$$

The nature of the shattering defining the split constraints depends on the set C, as we show in Section 5 and in the following example:

Example 6. *Consider Example 5: the constraint $V_2 \neq V_3$ can be shattered as $\{(\langle \{V_2\}, \{D_2\}, \{V_2 = i\} \rangle, \langle \{V_3\}, \{D_3\}, \{V_3 \neq i\} \rangle) \mid i \in D_2\}$. Each split i has a model count of* 2*, therefore summing out the subproblems from the shattering equals* 6*.*

5 Solving Combinatorial Problems

We show the efficacy of a lifted approach to counting constraint satisfaction problems by implementing a lifted solver, CoSo. CoSo solves problems over configurations $\{\cdot\}$ and $[\cdot]$ in CoLa, exploiting exchangeability and constraint shattering to lift the count. We define a set of operators that enables decomposing a combinatorial problem into subproblems where closed-form counting rules are applicable.

A CoLa problem is encoded in CoSo as a counting problem $MC(\mathcal{V}, D, C, (!\cdot))$. $\mathcal{V} = \{V_1, \ldots, V_n\}$ is a set of variables where n is the size of the configuration and $\mathcal{V}$ its name. When n is defined through inequalities, a different problem for each valid n is considered. $D = \{D_1, \ldots, D_n\}$ is the set of domains corresponding to each variable. For each $D_i \in D$, D_i is the universe, unless a positional constraint $\mathcal{V}[i] \in F$ is specified, then $D_i = F$. C is the set of additional size constraints and $(!\cdot)$ is one of $[|\cdot], [\|\cdot], \{|\cdot\}, \{\|\cdot\}$.

Example 7. *The CoLa encoding of **P3** (Example 3) is mapped to $MC(\mathcal{V}, D, C, \{|\cdot\})$: $\mathcal{V} = \{V_1, \ldots, V_5\}$, $D_i = \{s_1, \ldots, s_{12}\}$, $i \in \{1, \ldots, 5\}$,$C = \{\#(\{s_1, s_2, s_3\} \cap \mathcal{V}) \geqslant 2\}$.*

CoSo decomposes complex problems into liftable subproblems by shattering the constraints according to a set of operators. The preconditions of the operators distinguish between A) exchangeable ($\equiv^{\mathcal{V}}$) and B) non-exchangeable ($\not\equiv^{\mathcal{V}}$) variables. For the problems under investigation, variables are exchangeable if they have identical domains, i.e., they can take the same entities as values. In both cases we consider first 1) size constraints alone ($|C| \geqslant 1, (\|\cdot\)$), then 2) *injectivity* alone ($|C| = 0, (|\cdot\)$) and finally 3) the combination of the two ($|C| \geqslant 1, (|\cdot\)$). We begin with describing the base cases, where there are no size constraints ($|C| = 0$). We refer to the domain of a set of exchangeable variables $\mathcal{V}$ as $dom(\mathcal{V})$.

Base case. The base cases occur when there are no additional size constraints and either all the variables $\mathcal{V}$ are exchangeable ($\equiv^{\mathcal{V}}, |C| = 0, (!\cdot)$), or they are independent ($\equiv^{\mathcal{V}} \vee \not\equiv^{\mathcal{V}}, |C| = 0, (\|\cdot)$). The former corresponds to the cases in Table 1, thus a closed-form counting rule (binomial coefficient, factorial,...) solves the problem. The latter corresponds to independent variables, hence the multiplication rule can be adopted. We now present the operators that allow to reduce a problem to these base cases when other preconditions apply.

5.1 A. Exchangeable Variables

Exchangeable variables allow to lift the count of solutions.

A1. Size constraints. Consider a set of exchangeable variables $\mathcal{V}$, and a size constraint $c : \#F \cap \mathcal{V} = n$. We focus on equality as the other operators derive from it. In order to constrain $\mathcal{V}$ to satisfy c it is sufficient to constrain n variables in $\mathcal{V}$ to belong to F. Here exchangeability and distinguishability are exploited: under distinguishability of positions ($[\cdot]$) there are $e = \binom{|\mathcal{V}|}{n}$ exchangeable choices of variables to propagate c.

Precondition: $(\equiv^{\mathcal{V}}, C = \{\#F \cap \mathcal{V} = n\}, (\|\cdot))$
Operator: $\text{COUNT}_{\equiv}(\mathcal{V}, D, C, (\|\cdot)) =$
$D_{sat} = \{D_i = dom(\mathcal{V}) \cap F \mid i \in \{1, \dots, n\}\}$
$D_{unsat} = \{D_i = dom(\mathcal{V}) \cap \overline{F} \mid i \in \{n+1, \dots, |\mathcal{V}|\}\}$
$\mathbb{S}_1 = MC(\{V_1, \dots, V_n\}, D_{sat}, \{\}, (\|\cdot))$
$\mathbb{S}_2 = MC(\{V_{n+1}, \dots, V_{|\mathcal{V}|}\}, D_{unsat}, \{\}, (\|\cdot))$
Postcondition:
$(\mathbb{S}_1, \mathbb{S}_2)$ is a split for $\langle \mathcal{V}, D_{sat} \cup D_{unsat}, C, (\|\cdot)\rangle$
$MC(\mathcal{V}, D, C, (\|\cdot)) = e \cdot \mathbb{S}_1 \cdot \mathbb{S}_2$

After propagating c, the variables are no longer exchangeable: when $|C| > 1$ it is sufficient to propagate one of the size constraints and recursively solve the problem on non-exchangeable variables.

A2. and A3. As previously described, the count for $(|\cdot)$ over exchangeable variables (A2) is lifted with a counting rule (factorial). When both constraints are present (A3: $(\equiv^{\mathcal{V}}, |C| > 1, (|\cdot)))$, first a size constraint is propagated as in A1, then the problem is solved recursively.

5.2 B. Non-exchangeable Variables

When exchangeability cannot be exploited, we shatter the problem (and the constraints) into subproblems where variables are exchangeable and we exploit Property 3 to compute the model count of the constraint shattering. Consider the equivalence classes induced by the exchangeability relation $\equiv$ over $\mathcal{V}$: $\equiv$ induces a partition $\mathcal{V}/\equiv$ into sets (classes) of exchangeable variables ($[V] = \{X \in \mathcal{V} \mid X \equiv V\}$). Each class $[V]$ is thus a set of exchangeable variables: when $\mathcal{V}$ is not exchangeable $|\mathcal{V}/\equiv| > 1$. In order to shatter the problem we consider a split on a class $[V] \in \mathcal{V}/\equiv$ and the remaining variables $\hat{V} = \{X \in [W] | [W] \in \mathcal{V}/\equiv \setminus \{[V]\}\}$. Let $D_{[V]}$ and $D_{\hat{V}}$ be the respective sets of domains. The first subproblem is on exchangeable variables and can be solved accordingly, the second, on $\hat{V}$, can be further decomposed independently from $[V]$.

B1. Shattering size constraints. A size constraint $c : \#F \cap \mathcal{V} = n$ is shattered into the possible combinations of the number of elements belonging to F in the split class $[V]$ and the other variables in $\hat{V}$.

Precondition: $(\not\equiv^{\mathcal{V}}, C = \{\#F \cap \mathcal{V} = n\}, (\|\cdot))$
Operator: $\text{COUNT}_{\not\equiv}(\mathcal{V}, D, C, (\|\cdot)) =$
for i **in** $\{0, \dots, n\}$:
$c_{[V]} : \#F \cap [V] = i$
$c_{\hat{V}} : \#F \cap \hat{V} = n - i$
$\mathbb{S}^i_{[V]} = MC([V], D_{[V]}, \{c_{[V]}\}, (\|\cdot))$
$\mathbb{S}^i_{\hat{V}} = MC(\hat{V}, D_{\hat{V}}, \{c_{\hat{V}}\}, (\|\cdot))$
Postcondition:
$\{(\mathbb{S}^i_{[V]}, \mathbb{S}^i_{\hat{V}}) \mid i \in \{0, \dots, n\}\}$ shatters $\langle \mathcal{V}, D, C, (\|\cdot)\rangle$

Example 8. *Consider the following problem with universe* $U = F_1 \cup F_2$*:* $\mathcal{V} = \{V_1, \dots, V_4\}, D_1 = D_2 = D_4 = U, D_3 = F_2$ *and* $c : \#F_1 \cap \mathcal{V} \geqslant 2$*. Then* $[V_1] = \{V_1, V_2, V_4\}$*,* $dom([V_1]) = U$ *and* $\hat{V}_1 = [V_3] = \{V_3\}$*,* $dom([V_3]) = F_2$*. The shattering of* c *is:* $\{(\#F_1 \cap [V_1] = 0, \#F_1 \cap [V_3] \geqslant 2), (\#F_1 \cap [V_1] = 1, \#F_1 \cap [V_3] \geqslant 1), (\#F_1 \cap [V_1] \geqslant 2, \#F_1 \cap [V_3] \geqslant 0)\}$.

The shattering of multiple size constraints is obtained by recursively combining the individual constraint shattering with the splits of the other constraints. Let $\mathbb{S}_{\mathcal{V}} = MC(\mathcal{V}, D, C, (!\cdot))$ be a problem in a split and $c : \#F \cap \mathcal{V} = n$, then we denote by $\mathbb{S}_{\mathcal{V}} + c$ the problem $MC(\mathcal{V}, D, C \cup \{c\}, (!\cdot))$.

Precondition: $(\not\equiv^{\mathcal{V}}, |C| > 1, (\|\cdot))$
Operator: $\text{COUNT}_{\not\equiv}(\mathcal{V}, D, C, (\|\cdot))) =$
let $C = \{\#F \cap \mathcal{V} = n\} \cup C'$, $\mathbb{S}$ shatter $\langle \mathcal{V}, D, C', (\|\cdot)\rangle$:
for $(\mathbb{S}_{[V]}, \mathbb{S}_{\hat{V}}) \in \mathbb{S}$:
for $i \in \{0, \dots, n\}$:
$\mathbb{S}^i_{[V]} = \mathbb{S}_{[V]} + \{\#F \cap [V] = i\}$
$\mathbb{S}^i_{\hat{V}} = \mathbb{S}_{\hat{V}} + \{\#F \cap \hat{V} = n - i\}$
Postcondition:
$\{(\mathbb{S}^i_{[V]}, \mathbb{S}^i_{\hat{V}}) \mid i \in \{0, \dots, n\}\}$ shatters $\langle \mathcal{V}, D, C, (\|\cdot)\rangle$

B2. Shattering *injectivity*. The *injectivity* constraint is split by fixing a number n of entities in $[V]$ that allows to solve the problem on $\hat{V}$ knowing that n less different choices are available. Exchangeability can be exploited to avoid to reason on each possible entity choice, but rather on a set-level in terms of their size. Since an entity can have multiple properties, it may be the case that fine-grained information about the number specific properties of the choices in $[V]$ has to be set. $relevant(\hat{V})$ defines a partition of the universe with a granularity dependent on the sets (formulas) to be accounted for: $relevant(\{W\}) = \{W, \overline{W}\}$ and, recursively, $relevant(\hat{V} \cup \{Z\}) = \{Z \cap W | W \in relevant(\hat{V})\} \cup \{\overline{Z} \cap W | W \in relevant(\hat{V})\}$ (to avoid double counting when $|[\hat{V}]| = 1$ the base case is replaced by $relevant(\hat{V}) = \{\hat{V}\}$). An *histogram* associates each relevant set F to a cardinality: $histogram(\hat{V}) = \{(n, F) \mid F \in relevant(\hat{V})\}$ where $n \in \{0, \dots, |[V]|\}$ for each F. We denote by $histogram^*(\mathcal{V})$ the set of histograms corresponding to all possible choices of n for each relevant set. Note that relevant sets are a partition of

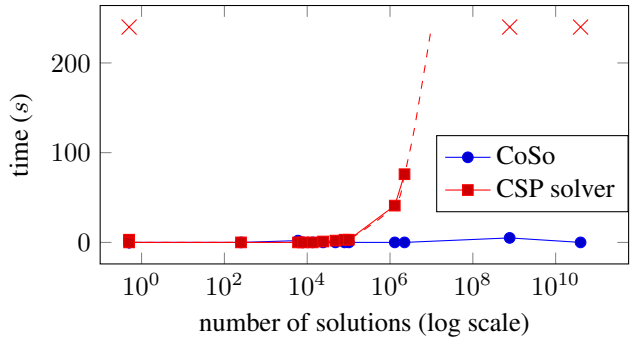

Figure 1: Solving time on synthetic benchmarks. Xs correspond to a timeout of $240s$

the domain, hence disjoint: considering a different n for any of them translates to a non-overlapping problem. Finally we denote with D - $histogram(\hat{V})$ the domain obtained from each $D_i \in D$ by removing a number of entities n from each corresponding relevant set (this can be also characterized by means of additional constraints over the admissible values).

Precondition: $(\not\equiv^{\mathcal{V}}, |C| = 0, (|\cdot|))$
Operator: $\text{ALLDIFF}_{\not\equiv}(\mathcal{V}, D, C, (|\cdot|))$ =
  for h **in** $histograms^*(\hat{V})$:
    $C_h = \{\#F \cap [V] = n \mid (n, F) \in h\}$
    $\mathbb{S}^h_{[V]} = MC([V], D_{[V]}, C_h, (|\cdot|))$
    $\mathbb{S}^h_{\hat{V}} = MC(\hat{V}, D_{\hat{V}} - h, C, (|\cdot|))$
Postcondition:
$\{(\mathbb{S}^h_{[V]}, \mathbb{S}^h_{\hat{V}}) \mid h \in histograms^*(\hat{V})\}$ shatters $\langle \mathcal{V}, D, C, (|\cdot|) \rangle$

Example 9. *Consider the setting of Example 8 where instead of c we shatter the injectivity constraint between $[V_1]$ and $[V_3]$: in $[V_1]$ up to 3 entities in F_1 can be selected. For clarity, we list only the pairs of size constraint and domain for the problem respectively on $[V_1]$ and $[V_3]$:* $\{(\{\#F_2 \cap [V_1] = i\}, dom(V_3) - \{(i, F_2)\}) \mid i \in \{0, \ldots, 3\}\}$

B3. Shattering both. When both injectivity and size constraints need to be split $(\not\equiv^{\mathcal{V}}, |C| > 1, (|\cdot|))$ the cases from the two shatterings are combined together, similarly to the case $(\not\equiv^{\mathcal{V}}, |C| > 1, (\|\cdot\|))$. We give an informal description since the shattering cases are the same for injectivity and size constraints as previously described: the size constraints from one shattering are considered in a Cartesian product-like combination with the cases from the other constraint shattering.

6 Experiments

We empirically evaluate two aspects of our contributions: (Q1) Is CoLa expressive enough to encode standard combinatorics math problems? (Q2) Does CoSo improve the solving efficiency compared to a traditional non-lifted framework?

We answer Q1 by analyzing a benchmark of natural language probability and combinatorics problems that were collected from online sources and textbooks [Dries *et al.*, 2017] and seeing if it is possible to encode the combinatorics problems in CoLa. It is possible to encode 88% (184 out of 210) of the benchmark's problems in CoLa, which provides some evidence that CoLa is expressive enough to represent combinatorics problems that arise in practice. The two most frequent limitations preventing an encoding are (1) constraints relating the position of groups of entities (e.g. "In how many ways can 5 boys and 5 girls be seated around a table so that no 2 boys sit next to each other?") and (2) mathematical constraints over choices (e.g. "Three distinguishable dice are thrown. In how many ways can they land and give a sum of 9?").

We answer Q2 by comparing CoSo with a non-lifted framework: MiniZinc [Nethercote *et al.*, 2007] paired with the solver Gecode [Schulte *et al.*, 2010]. The majority of the problems from [Dries *et al.*, 2017] involve a small number of entities and very simple constraints. Therefore, to highlight the benefits of lifted reasoning we consider a synthetic benchmark consisting of (randomly generated) #CSPs problems of the types $[|\cdot], [\|\cdot]$ and $\{|\cdot\}$ ($\{\|\cdot\}$ is not supported by MiniZinc). To increase the difficulty of the problems, we vary the number of entities (10, 15 and 20) and the size of the configurations (5, 10 and 15). Within each problem, two random subsets of entities (s_1, s_2) are defined and used in arbitrary constraints (e.g. $a[n_1] \in F_1 \cap F_2$, $\#(\overline{F_1} \cap a) > n_2$, where natural numbers n are also chosen randomly among meaningful values). An example of a problem in the benchmark is:

Example 10. *Synthetic benchmark example.*
$uni = \{e_1, \ldots, e_{20}\}$; $dom_2 = \{e_4, \ldots, e_{17}\}$;
$dom_1 = \{e_8, \ldots, e_{13}\}$; $a = [\,|\,uni]$; $\#a = 15$;
$a[1] \in dom_2 \cup \overline{dom_1}$; $a[3] \in dom_2$; $a[5] \in \overline{dom_2}$;
$\#(dom_2 \cap a) \leqslant 4$; $\#(dom_2 \cap a) \geqslant 2$;

Because increasing the number of entities and size as of the configuration results in larger search spaces, we consider the number of solutions as a proxy of problem difficulty. Figure 1 shows the solving time (y axis) w.r.t. the number of solutions (x axis, log-scale), where the X marks denote that a method exceeded the 240 second time limit. CoSo solves efficiently all the benchmarks proving the efficacy of lifted reasoning on combinatorics math problems. In contrast, the non-lifted CSP approach degrades exponentially as the number of solutions increases. Moreover, on the instance of type $[|\cdot]$ with 20 entities and configuration of size 15 (Example 10), MiniZinc times out even if the constraints are unsatisfiable (0 solutions). This showing how domain-lifted reasoning scales with an increasing number of entities regardless of the actual number of solutions. Hence, we can positively answer Q2.

7 Conclusion

We have presented a framework for combinatorial problems where existing algorithms and tools fail to provide an efficient solving strategy. We have introduced a declarative language, CoLa, that is able to represent a wide range of such problems, which we demonstrated by encoding a dataset of combinatorics problems from real world sources. We have developed new lifted inference techniques for this counting constraint satisfaction setting, have implemented a solver, CoSo, for a class of combinatorial problems, and have demonstrated that CoSo outperforms traditional solving techniques.

References

[Chesani *et al.*, 2017] Federico Chesani, Paola Mello, and Michela Milano. Solving mathematical puzzles: A challenging competition for AI. *AI Mag.*, 38(3):83–96, 2017.

[de Salvo Braz *et al.*, 2005] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth. Lifted first-order probabilistic inference. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pages 1319–1325. Professional Book Center, 2005.

[Dries *et al.*, 2017] Anton Dries, Angelika Kimmig, Jesse Davis, Vaishak Belle, and Luc De Raedt. Solving probability problems in natural language. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 3981–3987. ijcai.org, 2017.

[Kazemi *et al.*, 2016] Seyed Mehran Kazemi, Angelika Kimmig, Guy Van den Broeck, and David Poole. New liftable classes for first-order probabilistic inference. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 3117–3125, 2016.

[Kimmig *et al.*, 2017] Angelika Kimmig, Guy Van den Broeck, and Luc De Raedt. Algebraic model counting. *J. Appl. Log.*, 22:46–62, 2017.

[Milch *et al.*, 2008] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In Dieter Fox and Carla P. Gomes, editors, *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1062–1068. AAAI Press, 2008.

[Mitra and Baral, 2016] Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016.

[Nethercote *et al.*, 2007] Nicholas Nethercote, Peter J. Stuckey, Ralph Becket, Sebastian Brand, Gregory J. Duck, and Guido Tack. Minizinc: Towards a standard CP modelling language. In Christian Bessiere, editor, *Principles and Practice of Constraint Programming - CP 2007, 13th International Conference, CP 2007, Providence, RI, USA, September 23-27, 2007, Proceedings*, volume 4741 of *Lecture Notes in Computer Science*, pages 529–543. Springer, 2007.

[Niepert and Van den Broeck, 2014] Mathias Niepert and Guy Van den Broeck. Tractability through exchangeability: A new perspective on efficient probabilistic inference. In Carla E. Brodley and Peter Stone, editors, *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2467–2475. AAAI Press, 2014.

[Poole, 2003] David Poole. First-order probabilistic inference. In Georg Gottlob and Toby Walsh, editors, *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*, pages 985–991. Morgan Kaufmann, 2003.

[Roy and Roth, 2018] Subhro Roy and Dan Roth. Mapping to declarative knowledge for word problem solving. *Trans. Assoc. Comput. Linguistics*, 6:159–172, 2018.

[Schulte *et al.*, 2010] Christian Schulte, Guido Tack, and Mikael Z Lagerkvist. Modeling and programming with gecode. *Schulte, Christian and Tack, Guido and Lagerkvist, Mikael*, 1, 2010.

[Silver *et al.*, 2016] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–503, 2016.

[Stanley, 2012] Richard Stanley. *Enumerative combinatorics*. Cambridge University Press, New York, 2012.

[Taghipour *et al.*, 2012] Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel. Lifted variable elimination with arbitrary constraints. In Neil D. Lawrence and Mark A. Girolami, editors, *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, Spain, April 21-23, 2012*, volume 22 of *JMLR Proceedings*, pages 1194–1202. JMLR.org, 2012.

[Valiant, 1979] Leslie G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[Van den Broeck *et al.*, 2011] Guy Van den Broeck, Nima Taghipour, Wannes Meert, Jesse Davis, and Luc De Raedt. Lifted probabilistic inference by first-order knowledge compilation. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 2178–2185. IJCAI/AAAI, 2011.

[Van den Broeck *et al.*, 2014] Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In Chitta Baral, Giuseppe De Giacomo, and Thomas Eiter, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press, 2014.