Who did it?
Identifying the Most Likely Origins of Events

Marcel Gehrke[1] gehrke@ifis.uni-luebeck.de
Ralf Möller[1] moeller@uni-luebeck.de
Tanya Braun[2] tanya.braun@uni-muenster.de
[1] *Institute of Information Systems, University of Lübeck, Lübeck, Germany*
[2] *Data Science Group, University of Münster, Münster, Germany*

Abstract

One probabilistic inference task concerns answering queries for conditional marginal distributions, where a set of events is given. In this paper, we investigate the problem of only knowing that events are observed, from a number of sensors or for individuals, but not which sensors or individuals exhibit those events specifically. This situation might occur in multi-agent settings, such as in nanosystems, where single agents can no longer be tracked. However, to be able to perform probabilistic inference, those events need to be mapped to random variables, specifically to those that are most likely to exhibit those events. For the mapping, we show how lifting allows for generating all different possibilities to map those events, as we can do it over sets of indistinguishable random variables, leading to a set of queries. Given the mapping that leads to the most likely answer, we can construct evidence to perform probabilistic inference with. Finally, we compare solving the problem on the propositional level, which cannot be done in reasonable time, to our approach, which returns liftable evidence for tractable inference.

Keywords: Under-specified Evidence; Lifting; Query Answering; Probabilistic Inference.

1. Introduction

Query answering in probabilistic graphical models often means computing conditional marginal distributions of random variables (randvars) given observations for other randvars (events). For example, we may observe people attending a conference or having a paper accepted and ask for the probability of a researcher's reputation being excellent. However, a paper also requires that research has to be done on the topic of the paper, which we might not be able to observe directly, but which we can attribute to some parts of a research group. Consider nanosystems (Braun et al., 2021) for a more technical example: We cannot track each of the many agents and thus only observe that a task was performed a certain number of times but not by which agents specifically. So, what happens if we cannot observe the exact randvars for which events occurred, but only a number of events? In such a case, standard query answering is not possible as we cannot enter specific evidence in a model. Thus, this paper studies the problem of only knowing that an event has occurred a certain number of times, which implies that at least some randvars are interchangeable. We call this the *problem of under-specified evidence.*

Given under-specified evidence, the problem boils down to identifying those individuals that most likely exhibit the events in the model given all other (specified) evidence. Doing so on a propositional level is highly combinatorial and therefore not solveable in reasonable time as one has to consider assigning the number of events to different sets of randvars. With indistinguishable randvars for different individuals, i.e., graph symmetries, we may use lifting, which refers to efficiently handling sets of indistinguishable individuals using representatives (Poole, 2003), encoding symmetries using logical variables (logvars) with domains. As such, lifting has the potential to enable efficiently handling of under-specified evidence, allowing for tractability w.r.t. domain sizes Niepert and Van den Broeck (2014).

In general, lifted probabilistic inference leverages the relational aspect of a model, using representatives for groups of indistinguishable constants. Poole (2003) presents parametric factor (parfactor) graphs as relational models and proposes lifted variable elimination (LVE) as an exact inference algorithm on them. Taghipour et al. (2013b) extend LVE to its current form using an extensional constraint language for logvar. Algorithms such as the lifted junction tree algorithm (Braun and Möller, 2016) or first-order knowledge compilation (Van den Broeck et al., 2011) focus on efficient lifted inference for a set of queries. Braun and Möller (2019) consider the setting of all domains being unknown without evidence. Milch et al. (2005) investigate unknown objects and evidence for them. They look at a different problem, namely generating new constants for evidence, in contrast to the problem here, attributing evidence to existing constants.

Therefore, with the problem unsolved to the best of our knowledge, this paper formally defines the problem of under-specified evidence and introduces the Likely Constraints for Under-specified Evidence algorithm (CLUE) to solve the problem. The idea of CLUE is to identify all possible ways to use and reuse the sets of indistinguishable constants to match the cardinality of the under-specified events. Having all possibilities, CLUE asks, which possibility makes the under-specified evidence most likely in the model (multi-query answering). Based on the possibility yielding the most likely evidence, CLUE builds specific evidence, which can then be used with a lifted inference algorithm. When using liftable evidence only, CLUE also returns liftable evidence, allowing for continued tractable inference.

In the following, we recap parameterised probabilistic models (PMs) as a formalism for specifying probabilistic relational models, present CLUE, and discuss its implications.

2. Notation

We use PMs as a general formalism based on Poole (2003) and Taghipour et al. (2013b). PMs combine first-order logic with probabilistic models, using logvars as parameters in randvars to represent sets of indistinguishable randvars, forming parameterised randvars (PRVs). As an example, we set up a PM to model the reputation of researchers, inspired by the so-called competing workshop example (Milch et al., 2008), with a logvar representing researchers and journals respectively. A reputation is influenced by activities such as publishing, doing active research, and attending conferences.

Definition 1 (Logvar, PRV, Event) *Let* $\mathbf{R}$ *be a set of randvar names,* $\mathbf{L}$ *a set of logvar names,* Φ *a set of factor names, and* $\mathbf{D}$ *a set of constants. All sets are finite. Each logvar* L *has a domain* $\mathcal{D}(L) \subseteq \mathbf{D}$*. A* constraint *is a tuple* $(\mathcal{X}, C_{\mathcal{X}})$ *of a sequence of logvars* $\mathcal{X} = (X^1, \dots, X^n)$ *and a set* $C_{\mathcal{X}} \subseteq \times_{i=1}^n \mathcal{D}(X_i)$*. The symbol* $\top$ *for* C *marks that no restrictions apply, i.e.,* $C_{\mathcal{X}} = \times_{i=1}^n \mathcal{D}(X_i)$*. A* PRV $R(L_1, \dots, L_n), n \geq 0$ *is a syntactical*

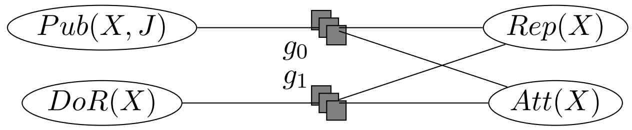

Figure 1: Parfactor graph for G^{ex}

construct of a randvar $R \in \mathbf{R}$ possibly combined with logvars $L_1, \ldots, L_n \in \mathbf{L}$. If $n = 0$, the PRV is parameterless and forms a propositional randvar. A PRV A (or logvar L) under constraint C is given by $A_{|C}$ ($L_{|C}$). We may omit $|\top$ in $A_{|\top}$ or $L_{|\top}$. The term $\mathcal{R}(A)$ denotes the possible values (range) of a PRV A. An event *$A = a$ denotes the occurrence of PRV A with range value $a \in \mathcal{R}(A)$.*

Consider $\mathbf{R} = \{Att, DoR, Rep, Pub\}$ for attends conference, does research, has a good reputation, and publishes in journals, respectively, and $\mathbf{L} = \{X, J\}$ with $\mathcal{D}(X) = \{x_1, x_2, x_3\}$ (people) and $\mathcal{D}(J) = \{j_1, j_2\}$ (journals), combined into Boolean PRVs $Att(X)$, $DoR(X)$, $Rep(X)$, and $Pub(X, J)$. A parfactor describes a function, for mapping argument values to real values (potentials).

Definition 2 (Parfactor, Model, Semantics) *We denote a* parfactor *g by $\phi(\mathcal{A})_{|C}$ with $\mathcal{A} = (A_1, \ldots, A_n)$ a sequence of PRVs, $\phi : \times_{i=1}^{n} \mathcal{R}(A_i) \mapsto \mathbb{R}^+$ a function with name $\phi \in \Phi$, and C a constraint on the logvars of $\mathcal{A}$. We may omit $|\top$ in $\phi(\mathcal{A})_{|\top}$. The term $lv(Y)$ refers to the logvars in some element Y, a PRV, a parfactor, or sets thereof. The term $gr(Y_{|C})$ denotes the set of all instances of Y w.r.t. constraint C. A set of parfactors $\{g_i\}_{i=1}^{n}$ forms a* PM *G. The semantics of G is given by grounding and building a full joint distribution. With Z as the normalisation constant, G represents $P_G = \frac{1}{Z} \prod_{f \in gr(G)} f$.*

Figure 1 shows a PM $G_{ex} = \{g_i\}_{i=0}^{1}$, with $g_0 = \phi_0(Rep(X), Att(X), Pub(X, J))_{|\top}$ and $g_1 = \phi_1(Rep(X), Att(X), DoR(X))_{|\top}$, each with eight input-output pairs (omitted).

Evidence displays symmetries when observing the same value for n instances of a PRV (Taghipour et al., 2013b). In a parfactor $g_E = \phi_E(R(\mathbf{X}))_{|C_E}$, a potential function ϕ_E and constraint C_E encode the observed values and instances for PRV $R(\mathbf{X})$. Assume we observe the value *true* for 10 randvars of the PRV $Rep(X)$. The corresponding parfactor is $\phi_E(Rep(X))_{|C_E}$. C_E represents the domain of X restricted to the 10 instances and $\phi_E(true) = 1$ and $\phi_E(false) = 0$. A technical remark: To *absorb* evidence, we split all parfactors g^i that cover $Rep(X)$, in a process called shattering (de Salvo Braz et al., 2005), restricting C_i to those tuples that contain $gr(Rep(X)_{|C_E})$ and a duplicate of g_i to the rest. The parfactor g_i absorbs g_E (see Taghipour et al. (2013b) for details).

Definition 3 (Query) *Given a PM G, a query term Q (ground PRV), and events $\mathbf{E} = \{E_i = e_i\}_{i=1}^{k}$, the expression $P(Q \mid \mathbf{E})$ denotes a* query *w.r.t. P_G.*

To answer a query using, e.g., LVE, LVE first absorbs evidence lifted and then eliminates all non-query terms. Lifting allows for eliminating a PRV $R(X)$ without grounding (conditions apply, Taghipour et al. (2013b)), which includes summing out $R(X)$ from the parfactor it occurs in as in propositional variable elimination, followed by taking the result to the power of the number of constants X represents in relation to the remaining constants.

The evidence just seen above is a case of specified evidence as we have the exact constants for which the evidence is observed in the constraint C_E. For under-specified evidence, the constraint is basically missing and as such, the parfactor is not complete.

3. Dealing with Under-specified Evidence

In standard lifted inference, we have to specify the constants for which an event is observed. Therefore, currently we cannot handle the information that there are 10 constants with $Rep = true$ without naming specific constants. We begin by defining this problem formally and then investigate how to solve it, before pouring our findings into CLUE.

3.1 The Problem of Under-specified Evidence

The problem deals with evidence for a specific PRV where the number of observations is known but the groundings are not. Since we aim for tractable query answering and evidence is only guaranteed to be liftable for PRVs with one logvar (Van den Broeck and Darwiche, 2013), we only consider evidence for PRVs with one logvar. We further assume that the model is preemptively shattered and the logvars are standardised apart for ease of exposition. We do not consider so-called counting randvars (CRVs) (Milch et al., 2008) as part of the initial model, i.e., only PRVs and classical propositional randvars occur. Those are standard assumptions for lifted inference. Additionally, we assume that logvars that have the same origin in terms of domain are identifiable. In such a model, we define under-specified evidence as follows:

Definition 4 (Under-specified Evidence) *Given a PRV $R(X)$, a range value $r \in \mathcal{R}(R)$, and a number $n, 0 < n \leq |\mathcal{D}(X)|$, indicating for how many groundings of the PRV the event $R = r$ is observed, we denote under-specified evidence by $(R(X) = r, n)$.*

The problem with under-specified evidence is that lifted inference algorithms cannot perform inference with such evidence as they need the constants for which the events are observed, i.e., specified evidence. Thus, on the technical side, the problem involves turning under-specified evidence into specified evidence by specifying a proper constraint C_E for an evidence parfactor $\phi_E(R(X))$ to then use standard lifted inference algorithms. For this transformation, we need to assign the evidence to some constants of the logvar domain, i.e., fill C_E with n constants from the domain of X. However, we do not want to assign random constants but rather those constants that are most likely to exhibit the observations. The formal definition of the problem follows:

Definition 5 (Problem of Under-specified Evidence) *Given a model G, some specified evidence $\mathbf{E}$, and under-specified evidence $(R_e(X_e) = r_e, n_e)_{e=1}^m$, the* problem of under-specified evidence *is to find those constants that make $(R_e(X_e) = r_e, n_e)_{e=1}^m$ most likely as specified evidence, i.e., with C a constraint on $X_1, \ldots, X_m$:*

$$\arg\max_C P(R_1(X_1) = r_1, \ldots, R_m(X_m) = r_m \mid \mathbf{E})_{|C}.$$

In terms of the semantics, we could ground the model, find all possible combinations of sets of randvars $R_e(x)$ of size n_e of all grounded randvars $gr(R_e(X))$, and test which combination is most likely in the grounded model. This course of action requires grounding, which we aim to avoid in lifted inference. Additionally, in lifted inference, we have the information about indistinguishability that we can use when finding the sets of size n_e, which means we have to test fewer combinations compared to the ground level. Thus, we next investigate how to solve this problem efficiently using lifting. We first look at finding the combinations and then we discuss answering the necessary queries.

3.2 Finding Possible Constraints for Under-specified Evidence

Let us first have a look at the example model G_{ex}, shown in Fig. 1, to illustrate the mechanics behind finding possible constraints given under-specified evidence.

Example 1 *Assume that* $|\mathcal{D}(X)| = 100$ *and remember that we have* $\top$ *constraints in* g_0 *and* g_1*, which means that the parfactors are valid for all of these* 100 *persons in the domain of* X*, i.e., the persons are indistinguishable. Given under-specified evidence* $(DoR(X) = true, 10)$ *and* $(Rep(X) = true, 10)$*, we have to identify those constants that are most likely to exhibit these observations. With* $\top$ *constraints, we have two possibilities, namely, that (i) the same* 10 *constants exhibit both observations at the same time (overlapping) and (ii) that* 10 *constants exhibit* $DoR(X) = true$ *and* 10 *other constants exhibit* $Rep(X) = true$ *(disjoint)* .

In this example, we have a model with $\top$ constraints and then observe two events for 10 constants each. Such a scenario is the ideal case for a lifted algorithm w.r.t. queries it needs to answer and therefore, also from a computational perspective. The reason lies in the fact that we only have to test two possibilities. As the original model has $\top$ constraints, we do not have to query $P(Rep(X'') = true, DoR(X') = true)$ as the result is identical to querying $P(Rep(X') = true, DoR(X'') = true)$. The reason lies in the constants behind X' and X'' being indistinguishable and therefore yielding the same distributions. Thus, we do not have to test all possible permutations, but only those that might lead to a different likelihood. On a propositional level, we actually would have to compute the queries for all possible permutation without the information of indistinguishability.

In case we observe events for 10 and 5 constants, the described procedure is also applicable. Here, one would also test one possibility with the 5 constants a subset of the 10 constants assigned to the other event or if they are a disjoint set. Then, we also just have to test the two possibilities. Thus, for how many constants we observe events does not change the basic idea if looking at $\top$ constraints. But what happens to the search space if we combine specified and under-specified evidence? To that end, we extend the previous example with specified evidence.

Example 2 *Assume that we observe that* $x_1, .., x_5$ *attend a conference and* $x_6, .., x_{10}$ *do not, which we enter as evidence into the example model* G_{ex} *using* absorption*. Absorption yields three versions of* g_0 *and* g_1 *each, namely, one version for the constants* $\{x_1, .., x_5\}$ *absorbing* $Att(X) = true$*, one for the constants* $\{x_6, .., x_{10}\}$ *absorbing* $Att(X) = false$*, and one for the remaining constants, i.e.,* $\phi_1^t(DoR(X^t), Rep(X^t))_{|C^t}$*,* $\phi_1^f(DoR(X^f), Rep(X^f))_{|C^f}$*, and* $\phi_1(Att(X), DoR(X), Rep(X))_{|C}$*, respectively, for* g_1 *with analogous versions for* g_0*. In this setting, we have to generate more combinations for the under-specified evidence of Example 1,* $(DoR(X) = true, 10)$ *and* $(Rep(X) = true, 10)$*.*

Overall, there are 19 *possibilities. The first case is to solely use the constants from* C^t *and* C^f *for both the* 10 *unspecified evidence instances. The second case of using* C^t *and the rest from* C *is more tricky. Here, we have the possibilities of:*

- $Rep(X^t) = true, DoR(X^t) = true, Rep(X') = true, DoR(X') = true$
  *(*5 *constants from* C^t *for both; same* 5 *constants from* C *for both),*
- $Rep(X^t) = true, DoR(X^t) = true, Rep(X') = true, DoR(X'') = true$
  *(*5 *from* C^t *for both; different* 5 *from* C *for each),*
- $Rep(X^t) = true, DoR(X'') = true, Rep(X') = true, DoR(X') = true$
  *(*5 *from* C^t *for Rep; same* 5 *from* C *for both and different* 5 *from* C *for DoR),*
- $Rep(X^t) = true, Rep(X') = true, DoR(X''') = true$
  *(*5 *from* C^t *and* 5 *from* C *for Rep, different* 10 *from* C *for DoR),*
- $Rep(X'') = true, DoR(X^t) = true, Rep(X') = true, DoR(X') = true$
  *(*5 *from* C^t *for DoR; same* 5 *from* C *for both and different* 5 *from* C *for Rep), and*
- $Rep(X''') = true, DoR(X^t) = true, DoR(X') = true$
  *(*5 *from* C^t *and* 5 *from* C *for DoR, different* 10 *from* C *for Rep).*

The third case for C^f *and the rest from* C *is analogous to* C^t *and* C*. The fourth case is to use* X^f, X^t*, and* X*. Here, we have the following possibilities:*

- $Rep(X^t) = true, DoR(X^f) = true, Rep(X') = true, DoR(X') = true$
  *(*5 *from* C^t *for Rep,* 5 *from* C^f *for Rep, same* 5 *from* C *for both),*
- $Rep(X^t) = true, DoR(X^f) = true, Rep(X') = true, DoR(X'') = true$
  *(*5 *from* C^t *for Rep,* 5 *from* C^f *for Rep, different* 5 *from* C *for each),*
- $Rep(X^f) = true, DoR(X^t) = true, Rep(X') = true, DoR(X') = true$
  *(*5 *from* C^f *for Rep,* 5 *from* C^t *for Rep, same* 5 *from* C *for both), and*
- $Rep(X^f) = true, DoR(X^t) = true, Rep(X') = true, DoR(X'') = true$
  *(*5 *from* C^f *for Rep,* 5 *from* C^t *for Rep, different* 5 *from* C *for each).*

The last case of only using X *is analogous to Example 1 with two possibilities, the same* 10 *constants for both or different* 10 *constants for each.*

Example 2 illustrates a few points worth noting: First, the number of possibilities depends on (i) the number of constraints per original logvar X in the model after absorbing specified evidence and (ii) the number m_x of under-specified evidence terms $(R_i(X) = r, n_i)_{i=1}^{m_x}$ per original logvar X. Second, to obtain the possible constraint sets, we have to check, for all possible combinations of existing constraints, how we can use them with and without reusing sets of their constants to fill up the under-specified evidence. More specifically, for a particular order of the under-specified evidence, we have to look at each constraint and see how we can use those constants with or without reusing them to fill the under-specified evidence in the given order in combination with all the other constraints and their ways of filling the under-specified evidence in that order. This procedure is to be repeated for all permutations of the order of the under-specified evidence. Third, such a problem can only be solved in reasonable time for a limited number of under-specified evidence PRVs as well as a limited number of sets of indistinguishable constants. In the example, the three constraints and two sets of under-specified evidence have already lead to $1 + 6 + 6 + 4 + 2 = 19$ possibilities. From the third point follows the last point, namely that solving this problem on a propositional level without exploiting indistinguishability leads to a deadly combinatorial explosion. The 19 possibilities in the lifted case is a drastic reduction to the propositional case, which would need to consider all possible subsets of cardinality 10 of the 100 grounded randvars of $Rep(X)$ and $DoR(X)$ each. Thus, using lifting enables problem instances to be solvable in reasonable time that otherwise would not be practically computable.

With the set of possibilities, each encoded as a constraint, the next steps are identifying the constraint that leads to the most likely evidence and using that constraint to build specified evidence out of the under-specified evidence.

3.3 Finding the Most Likely Constraint and Building Specified Evidence for It

The two steps of finding the most likely constraint and building the specified evidence based on that constraint are more straightforward compared to the previous step of finding all possible constraints. To illustrate the steps, let us come back to Example 1 and look at how we can compute how likely a constant is to exhibit a given event.

Example 3 *In continuation of Example 1 with $\top$ constraints and two sets of under-specified evidence terms, we query $P(Rep(X')_{|C'} = true, DoR(X')_{|C'} = true)$ and $P(Rep(X')_{|C'} = true, DoR(X'')_{|C''} = true)$ for the two possibilities that we have found. Given the two answers, we know now whether it is more likely that the identical 10 constants (from C' for both $Rep(X')$ and $DoR(X')$) or different 10 constants (from C' for $Rep(X')$ and from C'' for $DoR(X')$) exhibit the two event sets. Thus, we take the constraint that returns the higher probability for the query and construct specified evidence with it. Assuming that the setting with C' for both is the one with higher probability, we construct two evidence parfactors $\phi_E(DoR(X'))_{|C'}$ and $\phi_E(Rep(X'))_{|C'}$ with constraint C'.*

The queries in the example are actually parameterised queries (Braun and Möller, 2018) with logvars and constraints occurring in the query. Answering a parameterised query leads to CRVs in the result that encode how many of the represented groundings have assigned a specific range value, mapping to a probability. However, in the specific setting of queries over the same number of constants per constraint, we do not need to ask a parameterised query over all constants in the constraint but only a query for one representative constant in the constraint of the query term.[1] We can query representatives as the impact for each constant gets taken to the *same* power of the group size. Hence, if it is more likely for a representative, it will also be more likely for the whole group, which is also another upside to using lifting, which allows for representative (and parameterised) queries compared to a large conjunctive query in the ground case. In the example, this means that we do not have to ask the queries for all constants in C' and C'', but can query representatives for each group. If using a multi-query answering algorithm, the queries can be computed efficiently.

1. If CRVs occur in the model, we need parameterised queries. Here, we have excluded such a situation.

In Example 2, the sizes of the constraint sets differ with some over 5 constants and others over 10 constants. Thus, in this case, we cannot query representatives, but have to ask the parameterised query. As we need constraints that contain the same number of constants to be able to use representative queries, one could of course start to split up the constraints into sets of constants of equal cardinality, ask representative queries, and then combine the results according to the original constraints. In Example 2, we could split up the constraints over 10 constants into two constraints over 5 constants and then combine the results for those two. However, it is not always possible to split up constraints into sets of equal cardinality other than the trivial solution of cardinality 1. There is actually a trade-off between easier to compute queries and more constraint sets to handle during inference, which can lead to grounding in the first case as just argued.

Having illustrated how we can test which constants most likely exhibit an event, we present CLUE.

3.4 CLUE for Turning Under-specified Evidence into Specified Evidence

We have seen in the examples that there can be many possibilities for turning under-specified evidence into specified evidence. Further, to obtain all possibilities, CLUE has to test for many permutations how the sets of indistinguishable objects could be used. Algorithm 1 outlines our approach. Let us first have a look at the overall idea, before we go into detail. CLUE assumes liftable evidence, therefore it can go through all logvars one by one. Having all possibilities of how to use sets of indistinguishable objects for the under-specified evidence, CLUE still has to combine the possibilities across logvars, by taking the cross-product. Finally, CLUE computes which assignment leads to the highest probability and returns the corresponding evidence parfactors. Let us have a closer look at how finding all possible combinations works.

CLUE begins by absorbing evidence, which might change the sets of indistinguishable objects as we have seen in Example 2. Then, CLUE goes through each logvar L that has under-specified evidence. For L, CLUE begins by collecting all under-specified evidence terms, $\mathbf{E}_u^L$, concerning L and setting up a set for the queries for L. Next, CLUE goes through loops to construct the queries for L. First, CLUE iterates over all permutations p_e of $\mathbf{E}_u^L$, as CLUE later on fills the under-specified evidence in order, and by permuting $\mathbf{E}_u^L$, CLUE obtains all orders. Next, CLUE also permutes the constraints, as CLUE has to start with each constraint to fill up $\mathbf{E}_u^L$ along p_e. The next loop increments a number k that determines after how many under-specified evidence sets (k), CLUE moves on to the next constraint. The loop thereafter then ensures that after k PRVs, CLUE switches to the next constraint. Then, CLUE has one loop for how often a set of constants can be reused, j, and one loop for how many sets can be reuse, m, overall.

In the last loop, CLUE builds all the combinations given m, j, and the current position in the constraint permutation, cur. The function $p_{m,j,cur}$ builds all possibilities how to fill up the under-specified evidence, starting with the constraint at position cur with j constraints and up to m different sets being reused, using representative constants to be able to identify identical queries. For example with $j = 2$, $m = 2$, and $k = 2$, CLUE can use X^t and X^f both for DoR and Rep and fill up the under-specified evidence of our example completely. CLUE uses the constraint set C^t twice for Rep and DoR, so CLUE already has $j = 2$ but

Algorithm 1 Most Likely Constraints for Under-specified Evidence

function CLUE(Model G, Specified Evidence $\mathbf{E}_s$, Under-specified Evidence $\mathbf{E}_u$)
  if $\mathbf{E}_s \neq \emptyset$ **then**
    Absorb $\mathbf{E}_s$ in G ▷ Includes shattering G on $\mathbf{E}_s$
  for each logvar $L \in lv(G)$ with $L \in lv(\mathbf{E}_u)$ **do**
    $\mathbf{E}_u^L \leftarrow \{(R_l(L), n_l) \mid (R_l(L), n_l) \in \mathbf{E}_u\}$ ▷ Under-specified evidence for L
    $\mathbf{Q}^L \leftarrow \emptyset$ ▷ Query parts for L
    for each permutation p_e of $\mathbf{E}_u^L$ **do**
      for each permutation p_c of sets $C_{\mathcal{X}}, L \in \mathcal{X}$ in $\phi(\mathcal{A})_{|(\mathcal{X}, C_{\mathcal{X}})} \in G$ **do**
        for $k \leftarrow 1, \ldots, |\mathbf{E}_u^L|$ **do**
          Current constraint $cur \leftarrow 1$
          for $l = 1, l \leq |\mathbf{E}_u^L|, l+ = k$ **do**
            for $j \leftarrow 1, \ldots, k$ **do** ▷ how often to use a set of constants maximally
              for $m \leftarrow 1, \ldots, \frac{k \cdot |p_c|}{j}$ **do** ▷ how many sets of constants to reuse
                for each combination $p_{m,j,cur}$ using c constraints **do**
                  $C' \leftarrow$ Fill under-specified ev. l to $l + k - 1$ with $p_{m,j,cur}$
                  $cur \leftarrow cur + c + 1$
                  $\mathbf{Q}^L \leftarrow \mathbf{Q}^L \cup \{rv(\mathbf{E}_u^L)_{|C'}\}$
  $\mathbf{Q} \leftarrow \times_L \mathbf{Q}^L$
  $C_{\mathcal{X}} \leftarrow \arg\max_{C_{\mathcal{X}}} P(\mathbf{Q}_{|(\mathcal{X}, C_{\mathcal{X}})} = \mathbf{e}_u), \mathbf{Q}_{|(\mathcal{X}, C_{\mathcal{X}})} \in \mathbf{Q}$
  $\mathbf{E} \leftarrow$ Build specified evidence for $\mathbf{E}_u$ using $C_{\mathcal{X}}$
  return E

only used up 1 of the m's. The other m is then used for C^f to fill up *Rep* and *DoR*. With $j = 2$, $m = 1$, and $k = 2$, CLUE can again use C^t for *DoR* and *Rep*, but can only use C^f for either *DoR* or *Rep* and has to fill up the other PRV with X. With another permutation of constraints namely, C^t, C, C^f, CLUE would first use C^t for *DoR* and *Rep* and then C' for *DoR* and C'' for *Rep*. In such a fashion, CLUE constructs all possibilities. Unfortunately, to ensure that CLUE generates all possibilities, it needs all the loops mentioned before as well. However, with only limited numbers of under-specified evidence terms as well as constraints in the model as mentioned before, the loops do not need many iterations.

These combinations are then stored in $\mathbf{Q}^L$. After CLUE has computed all combinations for each logvar L, it computes the cross-product of the sets of queries for all logvars to obtain the queries it needs to ask. CLUE proceeds by asking the queries, selects the constraint with the highest probability and builds the corresponding evidence for it. Finally, CLUE returns the specified evidence for an inference algorithm to answer queries with it.

4. Discussion

The problem we set out to solve is how to identify the constants that most likely exhibit some event given some other events. CLUE uses the information of indistinguishable constants to solve the problem. The problem CLUE faces is to build all combinations with and without reusing sets of indistinguishable constants. To construct these possibilities, CLUE has to permute over the set of indistinguishable objects and the under-specified evidence PRVs as well as guarantee a few more restrictions to ensure that all possibilities are built. Overall, the process can be described as building few balls that fit the under-specified evidence given the sets of indistinguishable objects with reusing them. But is it possible to compute these possibilities in the lifted case without getting lost in the combinatorial problem?

Given many splits, the number of combinations increases as we have seen in the example. The number of possible combinations also increases with the number of PRVs for which CLUE has under-specified evidence. Nonetheless, we assume that there are groups behaving indistinguishably and hence, there will not be many different splits. Further, Van den Broeck and Darwiche (2013) show that given evidence for one-logvar PRVs, we can compute a lifted solution. In the average case, there will not be many splits, leading to just a few combinations to be queried. Normally, we expect at most 4 to 5 sets of indistinguishable objects. Further, we expect that there are at most 4 PRVs with under-specified evidence, as with graph symmetries, which are used for lifting, many randvars can be grouped together and it is unlikely that for many different PRVs, we cannot observe the origin of the events. Thus, the number can be bounded by these assumptions. Independent of the number of possibilities, the resulting specified evidence remains liftable evidence, which means that with a lifted inference algorithm and a liftable model and liftable queries, the inference problem remains practically computable.

As the combinatorial bounds depend on the under-specified evidence and the groups behaving indistinguishably, we provide over-approximated bounds for a specific example: Assuming we have under-specified evidence for 4 PRVs with n individuals each and we have 4 indistinguishable groups of size at least $4 \cdot n$. Thus, we can split each indistinguishable group in 4 parts of size n. Now, we can consider the 16 split indistinguishable groups as balls (κ) and the 4 PRVs as urns (ι). In this case, as we allow a ball to be used multiple times and we do not care about the order, we have $\binom{\kappa+\iota-1}{\iota-1} = 969$ possibilities. Of these possibilities, there are actually many leading to the very same result, e.g., using the first n instances of a indistinguishable group for all PRVs leads to the same result as using the second n instances from that group and so on, which we do not need to compute. In the ground case however, assuming $n = 10$, we have at least 160 balls (16 groups each of the size 10) to distribute on 10 urns for each under-specified evidence. Here, the case is slightly different, as while we fill up the urns for one under-specified evidence, we are not reusing the balls, but across under-specified evidence the balls can be reused. Thus, we have $\binom{\kappa}{\iota} = \binom{160}{10}$ possibilities, in the ground case for each under-specified evidence, leading to $4 \cdot \binom{160}{10}$. Here, increasing n would further drastically increase the number of possibilities, while the number stays the same in the lifted case. So, the number of balls and urns will always be significantly lower in the lifted case than in the ground case.

In some cases, CLUE can also save some computation time by only querying representatives. Further, there are inference engines using the lifted junction tree algorithm (LJT) to answer queries for the different combinations efficiently, the same holds for parameterised queries, for lifted models (Braun and Möller, 2018).

To solve such a problem on a propositional level and therefore, without the information of which constants are indistinguishable, one would have to test each combination of constants. Thus, we would have distinguishable balls of the number of our domain size, distinguishable urns for our different under-specified evidence, and for each of the urns an exact number of balls that have to be placed into. Further, each ball might be placed in each urn, but at most once into each urn. Solving such a problem is combinatorial and even for small domain sizes and just one or two under-specified evidences terms leads to very many possibilities. Even though also here the number of possibilities would be bounded, the bound would be really high for larger domain sizes. All of these possibilities would have to be checked and propositional inference is in the worst case exponential w.r.t. domain sizes (Taghipour et al., 2013a). Thus, this problem is intractable without the information, about which individuals are indistinguishable, in addition to lifted inference.

Overall, by using the knowledge of indistinguishable objects and preparing them well, CLUE solves our problem and does so in a rather efficient manner. Doing the same on a propositional level, we lose the information of indistinguishability and, therefore, have to check too many combinations to solve the given problem in all but very simple cases.

5. Conclusion

In this paper, we present CLUE to identify those individuals that most likely exhibit an event given a model. CLUE uses the information of indistinguishable objects of a lifted model to solve the problem, as this drastically reduces the search space, allowing for solving this problem for reasonable parameters. To build all possible combinations using the information of indistinguishability, CLUE goes through all permutations of PRVs of under-specified evidence, all permutations of sets of indistinguishable objects, all possibilities of reusing sets of indistinguishable objects, and how often sets of indistinguishable objects are changed. For those possibilities, CLUE asks the corresponding parameterised queries and builds for that assignment that has the highest probability the corresponding evidence parfactors. We also argue that doing so on a propositional level is not feasible.

The next step in this work involves finding a reasonable heuristics for guiding the search in the space of possibilities in contrast to generating all possibilities. Beyond that, an interesting future direction of this line of work lies in temporal probabilistic models, where we might get under-specified evidence terms for each time step. Given new evidence, it might turn out that actually another assignment is now more likely for some previous time step. Hence, the problem is slightly different as we want to find those individuals that most likely produced the event over time. Here, it might be an interesting idea to keep multiple assignments and prune the possibilities over time as we gain more information.

Acknowledgments

The research of MG was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy – EXC 2176 'Understanding Written Artefacts: Material, Interaction and Transmission in Manuscript Cultures', project no. 390893796. The research was conducted within the scope of the Centre for the Study of Manuscript Cultures (CSMC) at Universität Hamburg.

References

T. Braun and R. Möller. Lifted Junction Tree Algorithm. In *Proceedings of KI 2016: Advances in Artificial Intelligence*, pages 30–42. Springer, 2016.

T. Braun and R. Möller. Parameterised Queries and Lifted Query Answering. In *IJCAI-18 Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4980–4986. International Joint Conferences on Artificial Intelligence Organization, 2018.

T. Braun and R. Möller. Exploring Unknown Universes in Probabilistic Relational Models. In *Proceedings of AI 2019: Advances in Artificial Intelligence*, pages 487–500. Springer, 2019.

T. Braun, S. Fischer, F. Lau, and R. Möller. Lifting DecPOMDPs for Nanoscale Systems — A Work in Progress. In *10th International Workshop on Statistical Relational AI at the 1st International Joint Conference on Learning and Reasoning*, 2021.

R. de Salvo Braz, E. Amir, and D. Roth. Lifted First-order Probabilistic Inference. In *IJCAI05 Proceedings of the 19th International Joint Conference on Artificial intelligence*, pages 1319–1325. Morgan Kaufmann Publishers Inc., 2005.

B. Milch, B. Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. Blog: Probabilistic models with unknown objects. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1352–1359, 2005.

B. Milch, L. S. Zettlemoyer, K. Kersting, M. Haimes, and L. P. Kaelbling. Lifted Probabilistic Inference with Counting Formulas. In *AAAI08 Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, pages 1062–1068. AAAI Press, 2008.

M. Niepert and G. Van den Broeck. Tractability through Exchangeability: A New Perspective on Efficient Probabilistic Inference. In *AAAI14 Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2467–2475. AAAI Press, 2014.

D. Poole. First-order probabilistic inference. In *IJCAI03 Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 985–991. Morgan Kaufmann Publishers Inc., 2003.

N. Taghipour, J. Davis, and H. Blockeel. First-order Decomposition Trees. In *NIPS13 Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, pages 1052–1060. Curran Associates Inc., 2013a.

N. Taghipour, D. Fierens, J. Davis, and H. Blockeel. Lifted Variable Elimination: Decoupling the Operators from the Constraint Language. *Journal of Artificial Intelligence Research*, 47(1):393–439, 2013b.

G. Van den Broeck and A. Darwiche. On the Complexity and Approximation of Binary Evidence in Lifted Inference. In *NIPS13 Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, pages 2868–2876. Curran Associates Inc., 2013.

G. Van den Broeck, N. Taghipour, W. Meert, J. Davis, and L. De Raedt. Lifted Probabilistic Inference by First-order Knowledge Compilation. In *IJCAI11 Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, pages 2178–2185. AAAI Press/International Joint Conferences on Artificial Intelligence, 2011.