24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Lifting Queries for Lifted Inference (Abstract)[1]

Tanya Braun and **Ralf Möller** [2]

Abstract. Lifted algorithms use representatives for groups of indistinguishable objects to efficiently perform inference. Standard lifted algorithms like first-order variable elimination or first-order knowledge compilation, compute answers to marginal queries of *single* random variables or events in a lifted way using representatives. But, queries containing a *set of indistinguishable random variables* may lead to groundings, something that lifting tries to avoid. This paper presents parameterised queries as a means to avoid groundings, applying the lifting idea to queries. Parameterised queries enable lifted algorithms to compute answers faster, while compactly representing queries and answers.

1 Introduction

AI areas such as natural language understanding and machine learning need efficient inference algorithms. Modeling realistic scenarios yields large probabilistic models, requiring reasoning about sets of individuals. Lifting uses symmetries in a model to speed up reasoning with known domain objects. We study reasoning in large models that exhibit symmetries. Our inputs are a model and queries for probability distributions of random variables (randvars) given evidence. Inference tasks reduce to computing marginal distributions. Lifted variable elimination (LVE) allows for computing an answer to a query, lifting as many computations as possible [4].

LVE realises lifted inference using logical variables (logvars) as parameters to represent sets of interchangeable randvars, called parameterised randvars (PRVs). Consider an epidemic with many people possibly being sick, which we could model using a randvar E for "epidemic" and a PRV $S(X)$ for "sick", with logvar X representing a group of people. A parametric factor (parfactor) describes a function with PRVs as arguments that maps argument values to real values (potentials), identical for all argument groundings. An example parfactor is $\phi(E, S(X))$ describing the influence between E and $S(X)$. Still, queries concern single randvars, e.g., $S(eve)$ with eve being one of the people. A first step to answering a query is to preemptively shatter the model on the query, i.e., to split the logvars w.r.t. the randvars in the query [2]. Thus, a conjunctive query over a set of interchangeable objects such as $S(alice), S(eve), S(bob)$ leads to a grounding of the affected logvars, here, X.

To avoid groundings, we present *parameterised queries*, allowing logvars in query terms, and adapt LVE to them. With parameterised queries, we compute answers more efficiently and provide compact representations for queries and answers, possibly without any blow up. Parameterised queries come in handy in various scenarios: During an epidemic or a network attack, queries occur for how many people are likely sick or network components compromised. Let us look at an example showcasing the grounding problem.

[1] This paper is an abstract of a paper presented at IJCAI 2018 [1]

[2] University of Lübeck, Germany, email: <surname>@ifis.uni-luebeck.de

Example 1. Assume that the domain of X is $\{alice, eve, bob\}$. Given a parfactor $\phi(E, S(X), T(X))$ with another PRV $T(X)$ for people travelling and a query $P(S(alice), S(eve), S(bob))$, LVE shatters ϕ, which leads to effectively grounding X, with ϕ appearing in three versions, one for each domain value. Next, LVE eliminates the $Travel$ randvars, each elimination a copy of the other. To eliminate E, LVE multiplies the remaining parfactors into one parfactor with arguments E, $S(alice)$, $S(eve)$, and $S(bob)$, which means a size of $2^4 = 16$. LVE eliminates E from this product and normalises the result. The result parfactor $\phi'(S(alice), S(eve), S(bob))$ has $2^3 = 8$ mappings, which is exponential in the number of query terms. Table 1a shows the mappings, without explicit potentials as they depend on the concrete potentials in ϕ:

Table 1: Potential encodings

(a) Enumeration

$S(alice)$	$S(eve)$	$S(bob)$	ϕ'
$false$	$false$	$false$	v_0
$\boldsymbol{false}$	$\boldsymbol{false}$	$\underline{true}$	$\underline{\boldsymbol{v_1}}$
$\boldsymbol{false}$	$\underline{true}$	$\boldsymbol{false}$	$\underline{\boldsymbol{v_1}}$
$false$	$true$	$true$	v_2
$\underline{true}$	$\boldsymbol{false}$	$\boldsymbol{false}$	$\underline{\boldsymbol{v_1}}$
$true$	$false$	$true$	v_2
$true$	$true$	$false$	v_2
$true$	$true$	$true$	v_3

(b) Histogram

$\#_X[S(X)]$	ϕ''
$[0, 3]$	v_0
$[\underline{1}, \mathbf{2}]$	$\underline{\boldsymbol{v_1}}$
$[2, 1]$	v_2
$[3, 0]$	v_3

The potentials in Table 1a exhibit a symmetry: Two times a $\boldsymbol{false}$ value and one time a $\underline{true}$ value map to $\underline{\boldsymbol{v_1}}$. It is irrelevant whether $alice$, eve, or bob is the one being sick, as long as one has the value $\underline{true}$ assigned. The same holds for $false$ assigned once and $true$ assigned twice mapping to v_2. This symmetry is well-known in LVE: It has been used for counting [3]. A so-called counting randvar (CRV) encodes for n interchangeable randvars how many have a certain value. With a CRV $\#_X[S(X)]$ as input and histograms as values that specify for each value of $Sick(X)$ how many of the n randvars have this value, $\phi''(\#_X[S(X)])$ carries the same information as ϕ' (first position $S(X) = true$, second $S(X) = false$), shown in Table 1b.

The example illustrates three issues, (i) a large set of interchangeable query terms, (ii) inefficiencies during LVE, leading to (partial) groundings w.r.t. the referenced constants, to identical eliminations, and to large intermediate results, and (iii) a large result representation with symmetries. The example also shows that counting might help to counteract these issues. Specifically, we parameterise a query, which avoids shattering parfactors to ground instances and allows for using existing lifting techniques to enable a lifted calculation of queries over interchangeable query terms. As a corollary of lifted calculations, i.e., calculating a parameterised query without groundings, the result is compactly encoded using CRVs.

2 Compact Queries and Answers

To provide compact representations for queries as well as answers, we introduce parameterised queries, which also avoid groundings. A parameterised query may contain PRVs with logvars as query terms, achieving a compact query representation.

Definition 1. We denote *parameterised query terms* by $\mathbf{Q}_{|C}$, a constraint C denoting the groundings that are part of the query for the logvars in $\mathbf{Q}$, still allowing for grounding a PRV with a single domain value. To refer to complete domains, $\top$ is used.

Given parameterised queries, shattering no longer necessarily involves a grounding w.r.t. domain values appearing in a query. Instead, with a constraint that restricts the query PRVs to a subset of the instances in the model, shattering incurs a split for each query PRV in every parfactor that contains the query PRV. Of course, shattering may still result in a fine granularity in the model given multiple query PRVs and logvars appearing in various parfactors.

Example 2. $P(S(X)_{|\top})$ is an equivalent parameterised query for $P(S(alice), S(eve), S(bob))$. To ask for $P(S(alice), S(eve))$ without the grounding of *bob*, $P(S(X)_{|C})$ contains a constraint C referencing $alice$ and eve. Given $P(S(X)_{|\top})$, shattering does not change $\phi(E, S(X), T(X))$. Given $P(S(X)_{|C}))$, shattering leads to two versions instead of three.

With CRVs, we also have a concept that allows for representing the result in a compact way as shown in Example 1. LVE has an operator *count-convert* that produces a CRV under certain preconditions [4].

3 Lifted Query Answering

LVE for a parameterised query has the same workflow as before with model G, query terms $\mathbf{Q}$, and evidence $\mathbf{E}$ as inputs: After shattering G on $\mathbf{E}$ and $\mathbf{Q}$, G absorbs $\mathbf{E}$ and LVE eliminates all non-query randvars in G. Then, count conversions for the remaining logvars in G provide a compact result representation. If the logvars are not count-convertible, i.e., do not fulfil the preconditions of *count-convert*, LVE transforms G with further operators [4] to enable *count-convert*. The final step is normalising the result to produce probabilities. For normalisation, LVE cannot simply divide the potentials by the sum of all potentials if CRVs are involved. A histogram h stands for $Mul(h)$ assignments, with $Mul(h)$ referring to a multinomial coefficient. Normalising a potential w_i in a mapping $h_i \mapsto w_i$ with m overall mappings results in a normalised potential v_i as follows

$$v_i = \frac{w_i}{\sum_{i=0}^{m-1} Mul(h_i) w_i} \tag{1}$$

The resulting v_i only add up to 1 if multiplying $Mul(h_i)$ with v_i $\forall i$.

Example 3. For the query $P(S(X)_{|\top})$ and parfactor $\phi(E, S(X), T(X))$, the result after eliminating $T(X)$ is a parfactor $\phi(E, S(X))$ (one elimination instead of three). To eliminate E, we count-convert $S(X)$, leading to a parfactor $\phi(E, \#_X[S(X)])$ with $2 \cot 4 = 8$ mappings (instead of $2^4 = 16$). Eliminating E yields a parfactor $\phi(\#_X[S(X)])$, containing 4 mappings instead of $2^3 = 8$,

$$[0,3] \mapsto w_0,\ [1,2] \mapsto w_1,\ [2,1] \mapsto w_2,\ [3,0] \mapsto w_3,$$

in which $[1,2]$ and $[2,1]$ stand for $\frac{(1+2)!}{1!\cdot 2!} = \frac{(2+1)!}{2!\cdot 1!} = 3$ assignments. The other two histograms represent one assignment. The normalised potentials are given by $\frac{w_i}{(w_0+3\cdot w_1+3\cdot w_2+w_3)}$, which is equal to the v_i in Example 1. Without E, $S(X)$ is still count-convertible in a parfactor $\phi(S(X))$ to create a compact representation.

A parameterised query does not necessarily yield a result containing CRVs for exactly the PRVs and their constraints in the query. The query constraint holds the instances of the query PRVs not to eliminate. While eliminating all non-query PRVs, the query PRVs in the model may be affected by operators rewriting constraints (splits, groundings). They may appear fully grounded in the result simply through the application of operators to compute a correct result.

Example 4. Consider $T(eve) = true$ as evidence. After evidence absorption, there are two parfactors: $\phi'(E, S(eve))$, having absorbed the evidence, and $\phi''(E, S(X'), T(X'))$, now constrained to $alice$ and bob. When answering $P(S(X)_{|\top})$, LVE produces a parfactor $\phi(\#_{X'}[S(X')], S(eve))$. With more than three domain values of X, the result may look like $\phi(\#_{X'}[S(X')], \#_{X''}[S(X'')])$.

PRVs appearing split in the result allow for identifying groups within a PRV, appearing through evidence, as described above, or other dynamics in the model. Such a result easily supports continued processing, e.g., regarding most likely assignments to query PRVs or further eliminations to compute probabilities for a group of interest.

4 Outlook on Further Results

Parameterised queries facilitate a reduced *runtime* if they do not require a grounding of its logvars as a first LVE operation. With immediate groundings, runtimes are nearly identical, depending slightly on evidence. In any other case, parameterised queries allow for faster runtimes, saving operations during shattering and as a result during query answering. Even with groundings, as many operations as possible are lifted before reverting to propositional calculations for affected logvars. Additional count conversions at the end are necessary to induce a joint distribution over all instances in the query.

A query may still induce groundings by blocking a reasonable elimination order. The reason lies in a precondition for lifted summing out of a PRV A: A has to contain all logvars in a parfactor. If PRVs to eliminate contain fewer logvars than a query PRV and no count conversion applies, grounding a logvar is necessary. The result of a query that induces groundings is still correct. Only, the query PRV may be grounded in the result. We have identified conditions for grounding and liftable queries. If the underlying model is liftable, the following holds:

Theorem 1 (Completeness)**.** Parameterised query terms with only one logvar per term and one set of constants per domain are liftable.

Theorem 2 (Complexity)**.** The complexity of LVE for liftable parameterised queries is polynomial in domain sizes.

Corollary 1. CRVs compactly represent the result of liftable queries.

The full paper can be found at [1].

REFERENCES

[1] Tanya Braun and Ralf Möller, 'Parameterised Queries and Lifted Query Answering', in *IJCAI-18 Proc. of the 27th International Joint Conference on AI*, pp. 4980–4986. IJCAI Organization, (2018).

[2] Rodrigo de Salvo Braz, Eyal Amir, and Dan Roth, 'Lifted First-order Probabilistic Inference', in *IJCAI-05 Proc. of the 19th International Joint Conference on AI*, pp. 1319–1325. IJCAI Organization, (2005).

[3] Brian Milch, Luke S. Zettelmoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling, 'Lifted Probabilistic Inference with Counting Formulas', in *AAAI-08 Proc. of the 23rd AAAI Conference on AI*, pp. 1062–1068. AAAI Press, (2008).

[4] Nima Taghipour, Daan Fierens, Jesse Davis, and Hendrik Blockeel, 'Lifted Variable Elimination: Decoupling the Operators from the Constraint Language', *Journal of AI Research*, **47**(1), 393–439, (2013).