First-order Decomposition Trees Supplementary Material

Nima Taghipour **Jesse Davis** **Hendrik Blockeel**
Department of Computer Science, KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium

Abstract

In this document, we provide proofs for the theorems and present a basic algorithm for construction of FO-dtrees for PLMs.

1 Proof of Theorem 1

Let us first recall the Theorem.

Theorem 1 *A (non-counted) FO-dtree has a lifted inference solution if its clusters only consist of (representative) randvars and* 1*-logvar PRVs. We call such an FO-dtree a* liftable *tree.*

Proof. Following the discussion in the paper, each subproblem arising during inference requires handling a parfactor involving the randvars and PRVs that appear at the cluster of the node. To prove that each of these problems are liftable (do not require us to ground the PRVs and deal with all their randvars directly), we need to show that the whole group of randvars in each cluster can be partitioned into m groups of interchangeable k-tuples of randvars, with m and k independent of the domain size. We prove this relying on the properties of counting randvars in PLMs, and the correctness of counting conversion in LVE [2, 3]. For simplicity, let us assume that there are no ground randvars in the cluster (the generalization to include ground randvars is trivial). Then the model can be written as a 1-logvar parfactor as follows:

$$\phi(P_{11}(X_{11}), \dots P_{1,n_1}(X_{1,n_1}), \dots, P_{m1}(X_{11}), \dots P_{m,n_1}(X_{m,n_m})) \,|\, C,$$

in which for each $i \in \{1, \dots, m\}$, all X_{ij} are logvars from a distinct domain D_i, and P_{ij} is an PRV containing such a logvar—note that for the same i some X_{ij} (and some P_{ij}) can have the same name, although the PRVs are distinct. Since no PRV contains more than one logvar we can count-convert all the logvars in this model. This merges all distinct PRVs $P_{ij}(X_i)$ into one counting randvar. As such, by applying counting conversion on all the logvars X_{ij} of domain D_i, we can rewrite in the model the group of PRVs $P_{i1}(X_{i1}), \dots, P_{i,n_i}(X_{i,n_i})$ into a counting randvar

$$\#_{X_i}[P'_{i1}(X_i), \dots, P'_{i,k_i}(X_i)]$$

where P'_{ij} are the distinct predicates among P_{ij}, that is:

$$\{P'_{ij}(X_i)\}_{j=1}^{k_i} = \bigcup_{j=1}^{n_i} P_{ij}(X_i)$$

After counting all the logvars the parfactor becomes of the form

$$\phi'\big(\#_{X_1}[P'_{11}(X_1), \dots, P'_{1,k_1}(X_1)], \dots, \#_{X_m}[P'_{m1}(X_m), \dots, P'_{m,k_m}(X_m)]\big)$$

This shows that the whole group of randvars in the model can be partitioned into m groups of interchangeable k-tuples of randvars– one group of tuples for each counting randvar. Note that here

both k and m are independent of the domain size of the logvars: (i) m is the number of distinct domains among the logvars, and (ii) k can be no larger than the number of PRVs with a co-domain logvar in the model, that is, $k \leq \max\{k_i\}_i \leq \max\{n_i\}_i$. It is straight-forward to show that this also holds in the general case of a parfactor involving both 1-logvar and ground randvars.

2 Proof of Theorem 2

Let us first recall the Theorem.

Theorem 2 *The complexity of lifted variable elimination for a counted liftable FO-dtree T is:*

$$O(n_T \cdot \log n \cdot \exp(w_g) \cdot n_{\#}^{(w_{\#} \cdot r_{\#})}),$$

where n_T is the number of nodes in T, $(w_g, w_{\#})$ is its lifted width, n (resp., $n_{\#}$) is the the largest domain size among its logvars (resp., counted logvars), and $r_{\#}$ is the largest range size among its tuples of counted randvars.

Proof. We prove the theorem by bounding the complexity of each lifted operation performed at each of the n_T nodes of the tree. First consider a lifted elimination performed at some node T'. The complexity of this operation is proportional to $|range(cluster(T'))|$, as it needs to deal with a parfactor involving the (counting) randvars in the cluster. Each cluster is a group $\mathcal{A} = \{A_1, A_2, \dots A_{w'_g}, \gamma_1, \gamma_2, \dots, \gamma_{w'_{\#}}\}$ of randvars A_i, and counting randvars $\gamma_i = \#_{X_i}[P_{i1}(X_i), \dots, P_{ik}(X_i)]$, where $w'_{\#} \leq w_{\#}$, and $w'_g \leq w_g$.Thus

$$|range(\mathcal{A})| = \big(\prod_i |range(A_i)|\big) \cdot \big(\prod_j |range(\gamma_j)|\big).$$

For the first product, we have

$$\prod_{i=1}^{w'_g} |range(A_i)| = O(\exp(w_g)).$$

Moreover, since for each counting randvar γ_i, $|range(\gamma_i)| = O(n_i^{r_i})$, where n_i is the domain size of X_i, and r_i is the range size of the tuples of PRVs inside γ_i, for the second product we have

$$\prod_{j=1}^{w'_{\#}} |range(\gamma_j)| = O((n_{\#}^{r_{\#}})^{w_{\#}}) = O(n_{\#}^{(w_{\#} \cdot r_{\#})})$$

These two show that

$$|range(\mathcal{A})| = O(\exp(w_g) \cdot n_{\#}^{(w_{\#} \cdot r_{\#})})$$

This is the complexity of each lifted elimination step. Build on this we compute the complexity of the other two lifted operations, aggregation and counting conversion. For each of the $|range(\mathcal{A})|$ entries in the parfactor, these two operations perform an exponentiation which has complexity $O(\log n)$, where n is the domain size of the logvar. As such, this has complexity $O(\log n \cdot \exp(w_g) \cdot n_{\#}^{(w_{\#} \cdot r_{\#})})$. Since there at most one of each operation performed at each of the n_T nodes, the complexity of entire inference is

$$O(n_T \cdot \log n \cdot \exp(w_g) \cdot n_{\#}^{(w_{\#} \cdot r_{\#})}).$$

3 Finding corresponding FO-dtrees

In this section, we provide a simple algorithm that given a model G constructs a corresponding FO-dtree. Our method works in a top-down manner according to a recursive decomposition of G using $DPGs$. We also briefly discuss possible extensions of this simple algorithm, which can transform it into a greedy algorithm for finding 'better' trees.

We construct the tree top-down according to a recursive decomposition of G, which also employs $DPGs$ (Algorithm 1). At the beginning we have a single root node T with model G. According to

a decomposition of G into $\{G_i\}_i$ we add the children T_i of T to the tree, and then recursively build each tree T_i for G_i. Under DPG nodes we represent only one instance of the children. DPGs allow us to decompose the model into partial groundings, and recursive application of this tool results in a ground model. This allows us to reduce the problem to finding a dtree for the ground model.

FO-dtree(G)
if G is ground
  return DTREE(G)
if $\exists \mathbf{X}$ that allows DPG
  $T_{\mathbf{X}} \leftarrow$ DPG-NODE($\mathbf{X}, \mathbf{x}, G$)
  $G_{\mathbf{x}} = \{G\theta | \theta \in \Theta_{\mathbf{x}}\}$
  T.ADDCHILD(**FO-Dtree**($G_{\mathbf{x}}$))
else:
  $T \leftarrow$ NEWNODE()
  choose logvars $\mathbf{X}$ that co-occur in G:
  (there is always at least one choice $\mathbf{X} = \{X_i\}$)
  $G_{\mathbf{X}} \leftarrow \{g | \mathbf{X} \in logvar(g)\}$
  $G_{\neg \mathbf{X}} \leftarrow G \setminus G_{\mathbf{X}}$
  T.ADDCHILDREN(**FO-Dtree**($G_{\mathbf{X}}$),**FO-Dtree**($G_{\neg \mathbf{X}}$))
return T

Algorithm 1: A simple algorithm for finding a corresponding FO-dtree.

Extension to a greedy method for finding FO-dtrees. The above is a simple algorithm that shows the existence of a FO-dtree for each model, by finding one possible FO-dtree. While it does not consider the quality of the found FO-dtree, it can be easily modified into an algorithm that greedily searches for better trees, by performing better DPGs. For this we need to make two changes in Algorithm 1: (1) rename the logvars such that the model allows for a DPG, instead of relying on the naming of logvars in the model, and (2) select among the possible DPGs based on some criteria.

The first change requires us to *align* the logvars in different parfactors before performing a DPG, that is to rename the logvars properly such that a subset of the logvars allow for DPG. This is a simple generalization of finding an alignment between two parfactors, which is employed in lifted multiplication. This change allows us to consider all possible DPGs of the model in our search, without being restricted by the naming of logvars in the model. A natural way is to begin with a model in which all logvars are standardized apart among parfactors.

The second change allows us to consider the quality of different DPGs for selection among them. Here we give a score to possible DPGs, which is a greedy measure of the quality of their decomposition. For instance, we can simply consider the cutset size of the decomposition, or the size of its resulting clusters. A straightforward measure is comparing the lifted width of the resulting nodes, which takes into account also the opportunities exploited by counting. These two changes should be naturally incorporated into one module, which considers possible logvar re-namings (alignments) that enable some DPG, measures the quality of the corresponding DPGs, and selects among them. Search for alignments can be guided by considering the properties of logvars in the model [1, 4], and our result about computing properties of FO-dtree nodes based on the properties of logvars.

References

[1] Abhay Jha, Vibhav Gogate, Alexandra Meliou, and Dan Suciu. Lifted inference seen from the other side : The tractable features. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS)*, pages 973–981. 2010.

[2] Brian Milch, Luke S. Zettlemoyer, Kristian Kersting, Michael Haimes, and Leslie Pack Kaelbling. Lifted probabilistic inference with counting formulas. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI)*, pages 1062–1608, 2008.

[3] Nima Taghipour and Jesse Davis. Generalized counting for lifted variable elimination. In *Proceedings of the 2nd International Workshop on Statistical Relational AI (StaRAI)*, 2012.

[4] Guy Van den Broeck. On the completeness of first-order knowledge compilation for lifted probabilistic inference. In *Proceedings of the 24th Annual Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1386–1394, 2011.