



Universität Hamburg  
DER FORSCHUNG | DER LEHRE | DER BILDUNG

CHAI

Humanities-Centered AI



Deutsches Forschungszentrum  
für Künstliche Intelligenz  
*German Research Center for  
Artificial Intelligence*

# StaRAI: From a Probabilistic Propositional Model to a Highly Compressed Probabilistic Relational Model

ECSQARU 2025 – Hagen, Germany

Marcel Gehrke<sup>1</sup>, Malte Luttermann<sup>2</sup>

<sup>1</sup>Institute for Humanities-Centered Artificial Intelligence, University of Hamburg, Germany

<sup>2</sup>German Research Center for Artificial Intelligence (DFKI), Lübeck, Germany

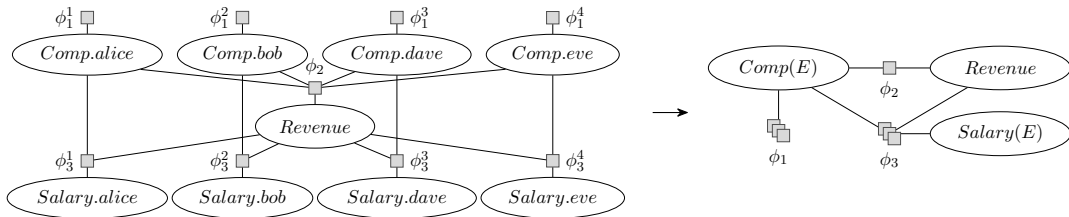
September 23, 2025

# Agenda

1. Introduction to relational models [Marcel]
2. Compressing probabilistic relational models [Malte]
  - ▶ Advancing the state of the art to obtain an exact compressed representation
  - ▶ Approximating a compressed representation with known error bounds
  - ▶ Handling unknown factors
3. Application: Lifted causal inference [Malte]
4. Summary [Marcel]

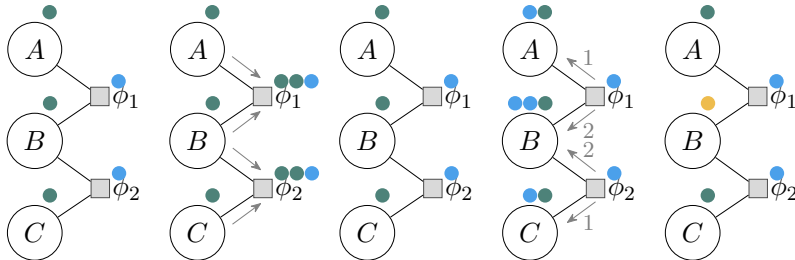
# Compressing Probabilistic Relational Models – Problem Setup

- ▶ Input: A factor graph  $G$
- ▶ Output: A parametric factor graph entailing equivalent semantics as  $G$ 
  - ▶ Under consideration of commutative factors
  - ▶ Independent of the order of factor's arguments



# Compressing Probabilistic Relational Models – Colour Passing

- Compression of a propositional model by passing colours around

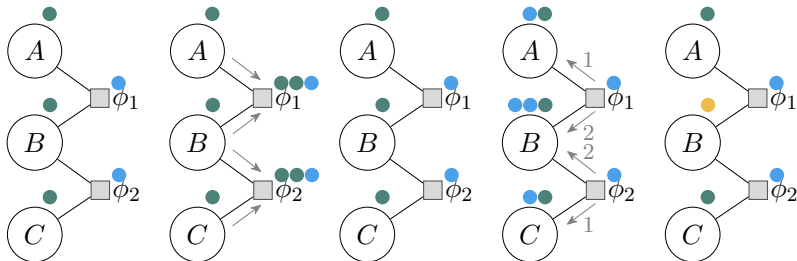


Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan (2009). »Counting Belief Propagation«. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-2009)*. AUAI Press, pp. 277–284.

Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan (2013). »Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training«. *Machine Learning* 92, pp. 91–132.

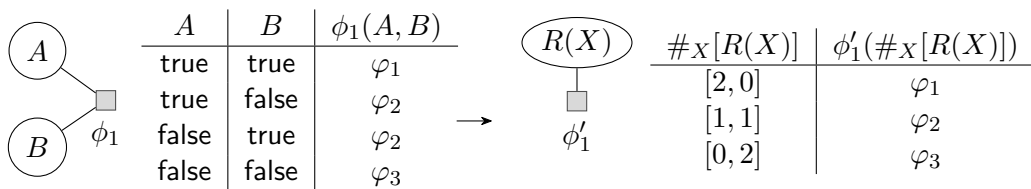
# Limitations of the Colour Passing Algorithm

1. Commutative factors are not recognised
2. Relies on fixed argument orders to detect equivalent factors
3. No logical variables are introduced



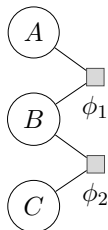
# Commutative Factors

- ▶ Detecting commutative factors allows to further compress the graph
- ▶ Histograms to efficiently detect and represent commutative factors
- ▶ Also applicable to a subset of arguments



# Permuted Factors

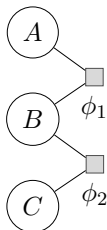
- ▶ Histograms are an efficient filter condition to check for the equivalence of factors
- ▶ Histograms determine possible permutations of factor arguments



A	B	$\phi_1(A, B)$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$

C	B	$\phi_2(C, B)$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$



A	B	$\phi_1(A, B)$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$

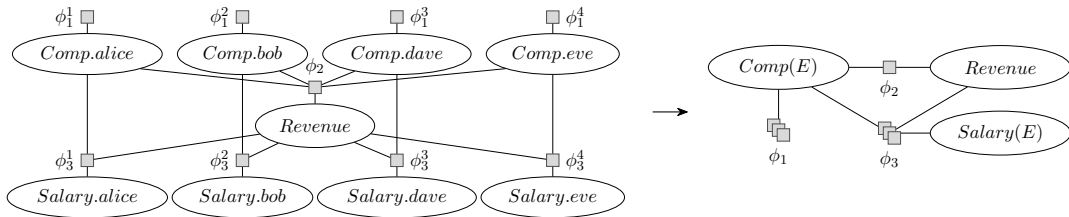
  

B	C	$\phi_2(B, C)$
true	true	$\varphi_1$
true	false	$\varphi_3$
false	true	$\varphi_2$
false	false	$\varphi_4$

# The Advanced Colour Passing Algorithm – Overview

Advanced Colour Passing (ACP) solves the limitations of the Colour Passing algorithm:

1. ACP recognises commutative factors
2. ACP detects equivalent factors independent of argument orders
3. ACP introduces logical variables

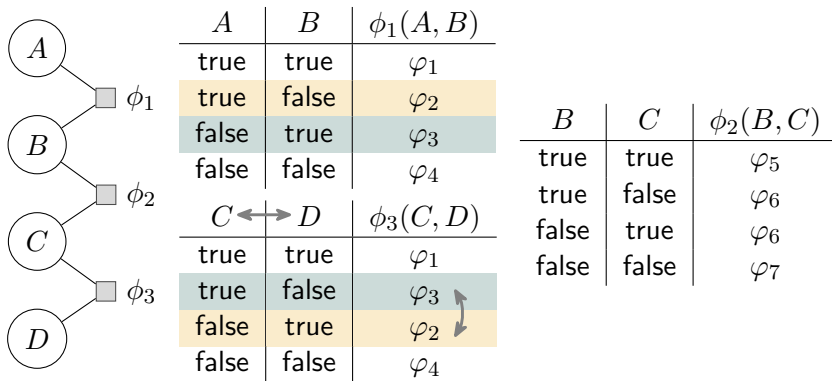


Malte Luttermann, Tanya Braun, Ralf Möller, and Marcel Gehrke (2024). »Colour Passing Revisited: Lifted Model Construction with Commutative Factors«. *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-2024)*. AAAI Press, pp. 20500–20507.



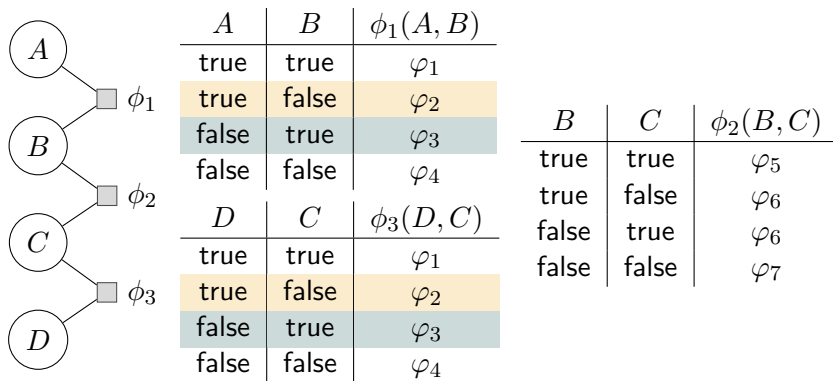
# The Advanced Colour Passing Algorithm – Example Run

1. Check for factor equivalence and rearrange arguments if necessary



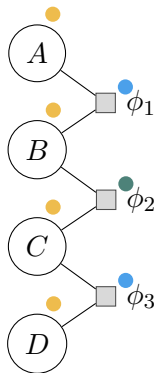
# The Advanced Colour Passing Algorithm – Example Run

1. Check for factor equivalence and rearrange arguments if necessary



# The Advanced Colour Passing Algorithm – Example Run

## 2. Assign initial colours

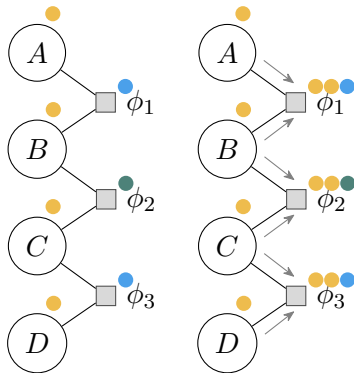


$A$	$B$	$\phi_1(A, B)$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$
$D$	$C$	$\phi_3(D, C)$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$

$B$	$C$	$\phi_2(B, C)$
true	true	$\varphi_5$
true	false	$\varphi_6$
false	true	$\varphi_6$
false	false	$\varphi_7$

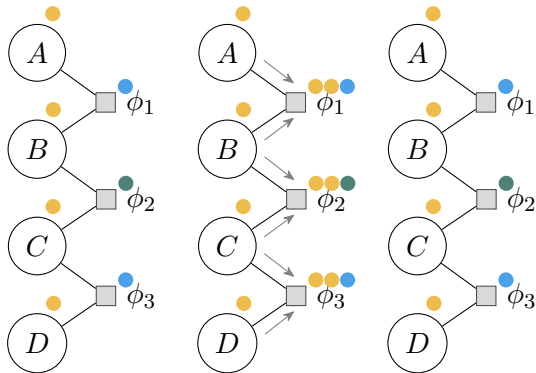
## The Advanced Colour Passing Algorithm – Example Run

### 3. Pass colours from variable nodes to factor nodes



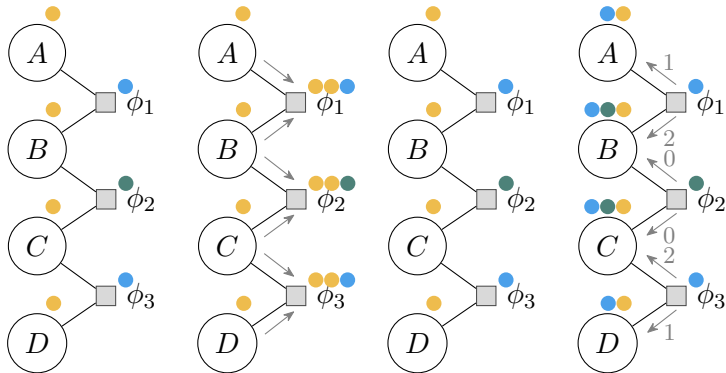
# The Advanced Colour Passing Algorithm – Example Run

## 4. Recolour factor nodes



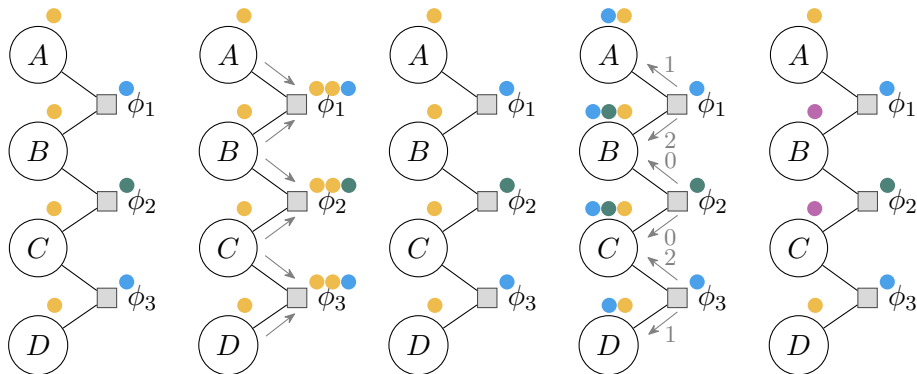
## The Advanced Colour Passing Algorithm – Example Run

5. Pass colours from factor nodes to variable nodes (omit positions if commutative)



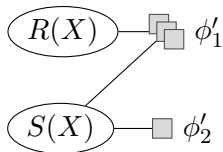
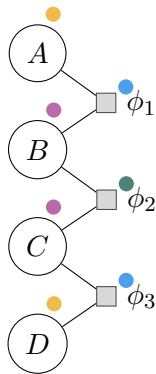
# The Advanced Colour Passing Algorithm – Example Run

## 6. Recolour variable nodes



# The Advanced Colour Passing Algorithm – Example Run

## 7. Introduce logical variables

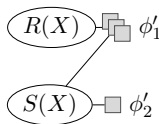
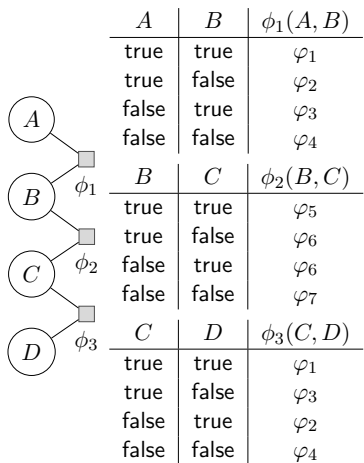


$R(X)$	$S(X)$	$\phi'_1(R(X), S(X))$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$

$\#_X[S(X)]$	$\phi'_2(\#_X[S(X)])$
$[2, 0]$	$\varphi_5$
$[1, 1]$	$\varphi_6$
$[0, 2]$	$\varphi_7$



# The Advanced Colour Passing Algorithm – Compression Result



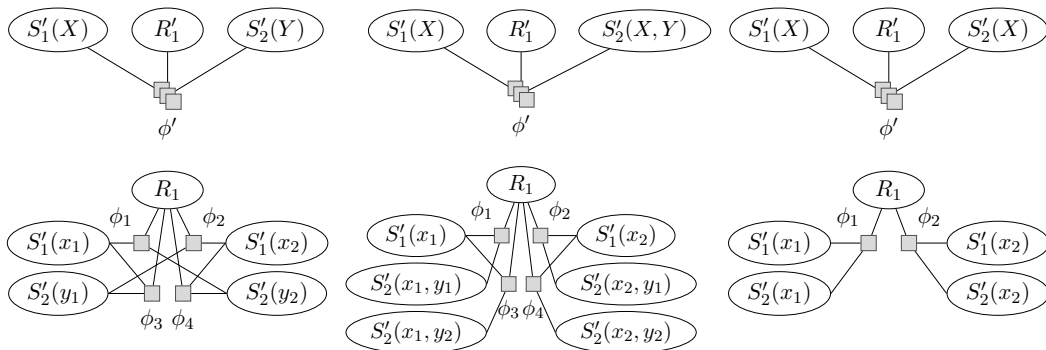
$R(X)$	$S(X)$	$\phi'_1(R(X), S(X))$
true	true	$\varphi_1$
true	false	$\varphi_2$
false	true	$\varphi_3$
false	false	$\varphi_4$

$\#_X[S(X)]$	$\phi'_2(\#_X[S(X)])$
[2, 0]	$\varphi_5$
[1, 1]	$\varphi_6$
[0, 2]	$\varphi_7$

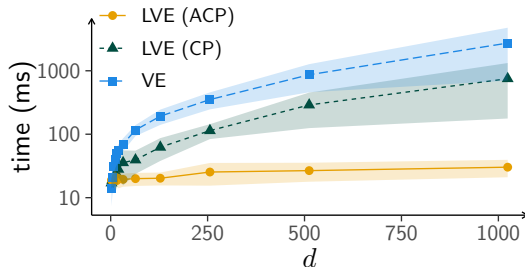
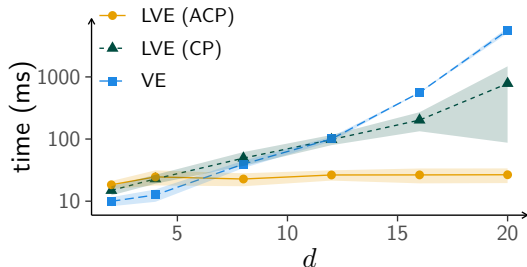
# Introduction of Logical Variables

- Introduction of logical variables to match groundings
- Three possible scenarios (distinct, binary, shared) in the domain-liftable fragment:



# The Advanced Colour Passing Algorithm – Experiments

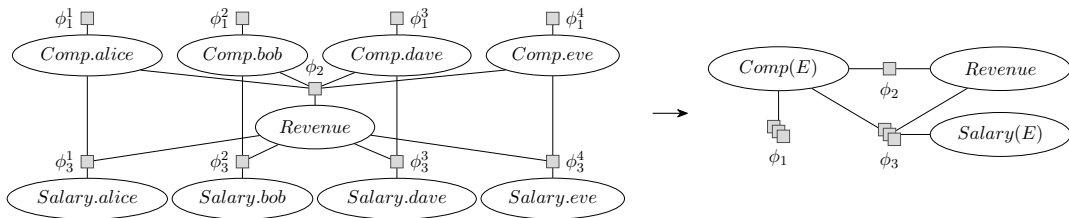
- ▶ Comparison of run times for lifted inference
- ▶ Left: Factor graphs with 1 commutative factor (no permuted argument orders)
- ▶ Right: Factor graphs where the arguments of 3 percent of the factors are permuted (no commutative factors)



# The Advanced Colour Passing Algorithm – Summary

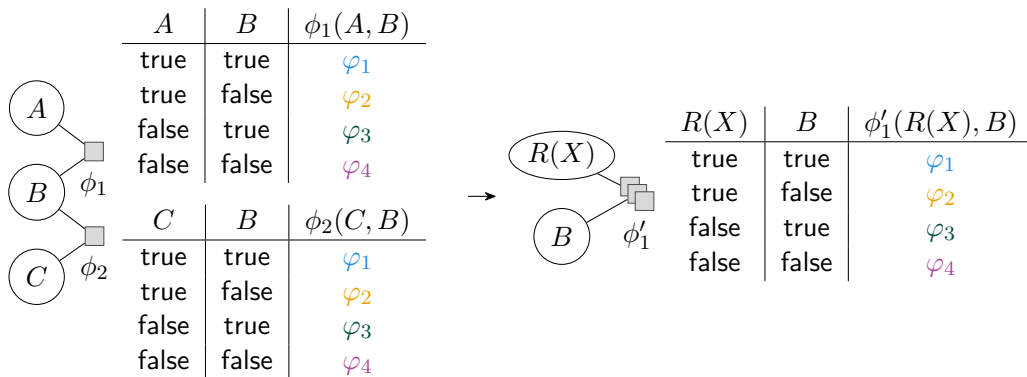
Limitations of Colour Passing solved:

1. Histograms to efficiently detect and compactly represent commutative factors
2. Histograms to efficiently identify factor equivalence
3. Logical variables for the class of domain-liftable models



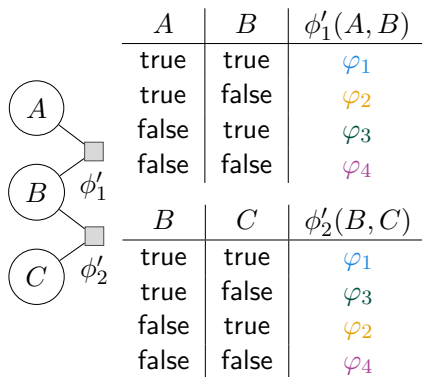
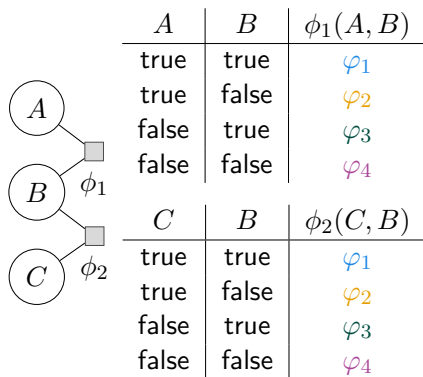
# Efficient Detection of Exchangeable Factors I

- Goal: Efficiently detect exchangeable (i.e., equivalent) factors
- Exchangeable factors encode an equivalent probability distribution



# Efficient Detection of Exchangeable Factors II

- Goal: Efficiently detect exchangeable (i.e., equivalent) factors
- Exchangeable factors encode an equivalent probability distribution



# Efficient Detection of Exchangeable Factors – Overview

Previously:

- ▶ Advanced Colour Passing algorithm to compress a factor graph
- ▶ *Buckets* to prune the search space, then iterating over all argument permutations
- ▶ Number of iterations in the worst-case for a factor with  $n$  arguments:  $O(n!)$

# Efficient Detection of Exchangeable Factors – Overview

Previously:

- ▶ Advanced Colour Passing algorithm to compress a factor graph
- ▶ *Buckets* to prune the search space, then iterating over all argument permutations
- ▶ Number of iterations in the worst-case for a factor with  $n$  arguments:  $O(n!)$

Next:

- ▶ Theoretical guarantees: *Buckets* to avoid iterating over permutations if possible
- ▶ Practical algorithm: Detection of Exchangeable Factors (DEFT) algorithm
- ▶ Number of iterations in the worst-case for a factor with  $n$  arguments:  $O(d)$  with  $d \ll n!$  in many practical settings

---

Malte Luttermann, Johann Machemer, and Marcel Gehrke (2024b). »Efficient Detection of Exchangeable Factors in Factor Graphs«. *Proceedings of the Thirty-Seventh International Florida Artificial Intelligence Research Society Conference (FLAIRS-2024)*. **Best Student Paper**. Florida Online Journals.



# Buckets

- Buckets count the occurrences of specific range values in an assignment
- Each potential belongs to *exactly one* bucket
- Each bucket contains *at least one* potential

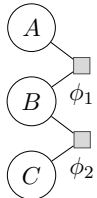


Diagram showing nodes  $A$ ,  $B$ , and  $C$ .  $A$  and  $B$  are connected by a square node labeled  $\phi_1$ .  $B$  and  $C$  are connected by a square node labeled  $\phi_2$ .

$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1(b)$	$\phi_2(b)$
$[2, 0]$	$\{\varphi_1\}$	$\{\varphi_1\}$
$[1, 1]$	$\{\varphi_2, \varphi_3\}$	$\{\varphi_3, \varphi_2\}$
$[0, 2]$	$\{\varphi_4\}$	$\{\varphi_4\}$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

# Properties of Buckets I

- If at least one bucket is mapped to different multisets of values by  $\phi_1$  and  $\phi_2$ , then  $\phi_1$  and  $\phi_2$  are not exchangeable (Luttermann, Braun, Möller, and Gehrke, 2024)

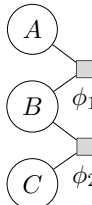


Diagram illustrating nodes A, B, and C, and their mappings  $\phi_1$  and  $\phi_2$ .

$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1(b)$	$\phi_2(b)$
$[2, 0]$	$\{\varphi_1\}$	$\{\varphi_1\}$
$[1, 1]$	$\{\varphi_2, \varphi_3\}$	$\{\varphi_3, \varphi_2\}$
$[0, 2]$	$\{\varphi_4\}$	$\{\varphi_4\}$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

## Properties of Buckets II

- Two factors are exchangeable iff there exists a permutation of their arguments such that each bucket is mapped to the same ordered multiset of values

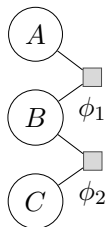
$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1^\gamma(b)$	$\phi_2^\gamma(b)$
$[2, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[1, 1]$	$\langle \varphi_2, \varphi_3 \rangle$	$\langle \varphi_3, \varphi_2 \rangle$
$[0, 2]$	$\langle \varphi_4 \rangle$	$\langle \varphi_4 \rangle$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

# The Detection of Exchangeable Factors Algorithm – Example Run

- Iterate over buckets and search for possible rearrangements to obtain identically ordered multisets within each bucket



$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

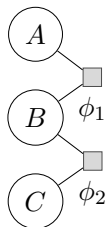
$b$	$\phi_1^\succ(b)$	$\phi_2^\succ(b)$
$[2, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[1, 1]$	$\langle \varphi_2, \varphi_3 \rangle$	$\langle \varphi_3, \varphi_2 \rangle$
$[0, 2]$	$\langle \varphi_4 \rangle$	$\langle \varphi_4 \rangle$

Possible rearrangements:

$[2, 0]: B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

# The Detection of Exchangeable Factors Algorithm – Example Run

- Iterate over buckets and search for possible rearrangements to obtain identically ordered multisets within each bucket



$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1^\succ(b)$	$\phi_2^\succ(b)$
$[2, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[1, 1]$	$\langle \varphi_2, \varphi_3 \rangle$	$\langle \varphi_3, \varphi_2 \rangle$
$[0, 2]$	$\langle \varphi_4 \rangle$	$\langle \varphi_4 \rangle$

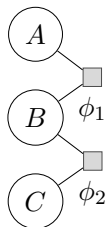
Possible rearrangements:

$[2, 0]: B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

$[1, 1]: B \rightarrow \{2\}, C \rightarrow \{1\}$

# The Detection of Exchangeable Factors Algorithm – Example Run

- Iterate over buckets and search for possible rearrangements to obtain identically ordered multisets within each bucket



$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1^\succ(b)$	$\phi_2^\succ(b)$
$[2, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[1, 1]$	$\langle \varphi_2, \varphi_3 \rangle$	$\langle \varphi_3, \varphi_2 \rangle$
$[0, 2]$	$\langle \varphi_4 \rangle$	$\langle \varphi_4 \rangle$

Possible rearrangements:

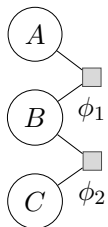
$[2, 0]$ :  $B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

$[1, 1]$ :  $B \rightarrow \{2\}, C \rightarrow \{1\}$

$[0, 2]$ :  $B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

# The Detection of Exchangeable Factors Algorithm – Example Run

- Iterate over buckets and search for possible rearrangements to obtain identically ordered multisets within each bucket



$A$	$B$	$\phi_1(A, B)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_2$	$[1, 1]$
false	true	$\varphi_3$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$B$	$C$	$\phi_2(B, C)$	$b$
true	true	$\varphi_1$	$[2, 0]$
true	false	$\varphi_3$	$[1, 1]$
false	true	$\varphi_2$	$[1, 1]$
false	false	$\varphi_4$	$[0, 2]$

$b$	$\phi_1^\succ(b)$	$\phi_2^\succ(b)$
$[2, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[1, 1]$	$\langle \varphi_2, \varphi_3 \rangle$	$\langle \varphi_3, \varphi_2 \rangle$
$[0, 2]$	$\langle \varphi_4 \rangle$	$\langle \varphi_4 \rangle$

Possible rearrangements:

$[2, 0]: B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

$[1, 1]: B \rightarrow \{2\}, C \rightarrow \{1\}$

$[0, 2]: B \rightarrow \{1, 2\}, C \rightarrow \{1, 2\}$

Intersection:  $B \rightarrow \{2\}, C \rightarrow \{1\}$

# Degree of Freedom (Theoretical Guarantees)

- ▶ Duplicate values in buckets increase complexity
- ▶ Degree of freedom of a bucket  $b$ :

$$\mathcal{F}(b) = \prod_{\varphi \in \text{unique}(\phi^{\succ}(b))} \text{count}(\phi^{\succ}(b), \varphi)!$$

- ▶ Degree of freedom of a factor  $\phi$ :

$$\mathcal{F}(\phi) = \min_{b \in \{b | b \in \mathcal{B}(\phi) \wedge |\phi^{\succ}(b)| > 1\}} \mathcal{F}(b)$$

$b$	$\phi_1'^{\succ}(b)$	$\phi_2'^{\succ}(b)$
$[3, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[2, 1]$	$\langle \varphi_2, \varphi_3, \varphi_5 \rangle$	$\langle \varphi_3, \varphi_5, \varphi_2 \rangle$
$[1, 2]$	$\langle \varphi_4, \varphi_6, \varphi_6 \rangle$	$\langle \varphi_6, \varphi_4, \varphi_6 \rangle$
$[0, 3]$	$\langle \varphi_7 \rangle$	$\langle \varphi_7 \rangle$



# Degree of Freedom (Theoretical Guarantees)

- ▶ Duplicate values in buckets increase complexity
- ▶ Degree of freedom of a bucket  $b$ :

$$\mathcal{F}(b) = \prod_{\varphi \in \text{unique}(\phi^\succ(b))} \text{count}(\phi^\succ(b), \varphi)!$$

- ▶ Degree of freedom of a factor  $\phi$ :

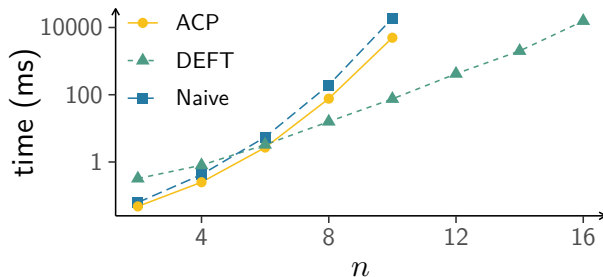
$$\mathcal{F}(\phi) = \min_{b \in \{b | b \in \mathcal{B}(\phi) \wedge |\phi^\succ(b)| > 1\}} \mathcal{F}(b)$$

$b$	$\phi_1'^\succ(b)$	$\phi_2'^\succ(b)$
$[3, 0]$	$\langle \varphi_1 \rangle$	$\langle \varphi_1 \rangle$
$[2, 1]$	$\langle \varphi_2, \varphi_3, \varphi_5 \rangle$	$\langle \varphi_3, \varphi_5, \varphi_2 \rangle$
$[1, 2]$	$\langle \varphi_4, \varphi_6, \varphi_6 \rangle$	$\langle \varphi_6, \varphi_4, \varphi_6 \rangle$
$[0, 3]$	$\langle \varphi_7 \rangle$	$\langle \varphi_7 \rangle$

- ▶ The number of table comparisons needed to check whether  $\phi_1$  and  $\phi_2$  are exchangeable is in  $O(d)$ , where  $d = \min\{\mathcal{F}(\phi_1), \mathcal{F}(\phi_2)\}$  (i.e.,  $d$  is factorial)
- ▶ In many practical settings it holds that  $d \ll n!$
- ▶ The degree of freedom is upper-bounded by  $n!$

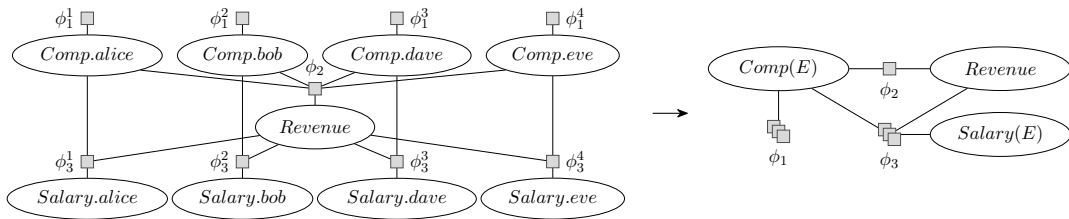
# The Detection of Exchangeable Factors Algorithm – Experiments

- ▶ Comparison of run times of DEFT, ACP (previous work), and a »naive« approach
- ▶ Average over exchangeable and non-exchangeable factors with a proportion of identical potentials in  $\{0.0, 0.1, 0.2, 0.5, 0.8, 0.9, 1.0\}$
- ▶ Timeout after 30 minutes per instance



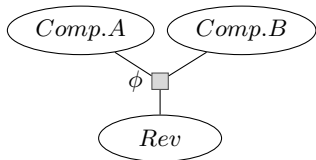
# The Detection of Exchangeable Factors Algorithm – Summary

- ▶ Problem of detecting exchangeable factors efficiently solved
- ▶ Upper bound on the number of table comparisons depending on the number of duplicate potential values within the buckets of the factors
- ▶ The DEFT algorithm exploits this upper bound effectively in practice



# Efficient Detection of Commutative Factors I

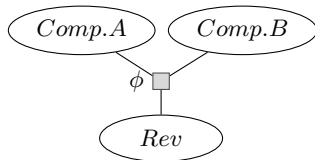
- ▶ Goal: Efficiently detect commutative factors
- ▶ Commutative factors have symmetries within themselves due to their arguments being exchangeable



$Comp.A$	$Comp.B$	$Rev$	$\phi(Comp.A, Comp.B, Rev)$
true	true	true	$\varphi_1$
true	true	false	$\varphi_4$
true	false	true	$\varphi_2$
true	false	false	$\varphi_5$
false	true	true	$\varphi_2$
false	true	false	$\varphi_5$
false	false	true	$\varphi_3$
false	false	false	$\varphi_6$

# Efficient Detection of Commutative Factors I

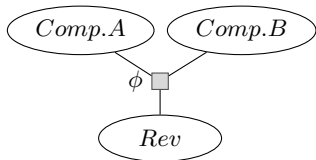
- ▶ Goal: Efficiently detect commutative factors
- ▶ Commutative factors have symmetries within themselves due to their arguments being exchangeable



<i>Comp.A</i>	<i>Comp.B</i>	<i>Rev</i>	$\phi(\textit{Comp.A}, \textit{Comp.B}, \textit{Rev})$
true	true	true	$\varphi_1$
true	true	false	$\varphi_4$
true	false	true	$\varphi_2$
true	false	false	$\varphi_5$
false	true	true	$\varphi_2$
false	true	false	$\varphi_5$
false	false	true	$\varphi_3$
false	false	false	$\varphi_6$

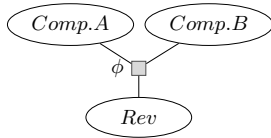
# Efficient Detection of Commutative Factors I

- ▶ Goal: Efficiently detect commutative factors
- ▶ Commutative factors have symmetries within themselves due to their arguments being exchangeable

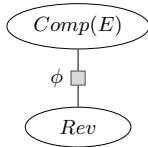


$Comp.A$	$Comp.B$	$Rev$	$\phi(Comp.A, Comp.B, Rev)$
true	true	true	$\varphi_1$
true	true	false	$\varphi_4$
true	false	true	$\varphi_2$
true	false	false	$\varphi_5$
false	true	true	$\varphi_2$
false	true	false	$\varphi_5$
false	false	true	$\varphi_3$
false	false	false	$\varphi_6$

# Efficient Detection of Commutative Factors II



$Comp.A$	$Comp.B$	$Rev$	$\phi(Comp.A, Comp.B, Rev)$
true	true	true	$\varphi_1$
true	true	false	$\varphi_4$
true	false	true	$\varphi_2$
true	false	false	$\varphi_5$
false	true	true	$\varphi_2$
false	true	false	$\varphi_5$
false	false	true	$\varphi_3$
false	false	false	$\varphi_6$



$\#_E[Comp(E)]$	$Rev$	$\phi(\#_E[Comp(E)], Rev)$
[2, 0]	true	$\varphi_1$
[1, 1]	true	$\varphi_2$
[0, 2]	true	$\varphi_3$
[2, 0]	false	$\varphi_4$
[1, 1]	false	$\varphi_5$
[0, 2]	false	$\varphi_6$

# Efficient Detection of Commutative Factors – Overview

Previously:

- ▶ Advanced Colour Passing algorithm to compress a factor graph
- ▶ Iteration over all subsets of arguments to find commutative arguments
- ▶ Number of iterations for a factor with  $n$  arguments is in  $O(2^n)$



# Efficient Detection of Commutative Factors – Overview

Previously:

- ▶ Advanced Colour Passing algorithm to compress a factor graph
- ▶ Iteration over all subsets of arguments to find commutative arguments
- ▶ Number of iterations for a factor with  $n$  arguments is in  $O(2^n)$

Next:

- ▶ Theoretical guarantees: *Buckets* to avoid iterating over all subsets
- ▶ Practical algorithm: Detection of Commutative Factors (DECOR) algorithm

---

Malte Luttermann, Johann Machemer, and Marcel Gehrke (2024a). »Efficient Detection of Commutative Factors in Factor Graphs«. *Proceedings of the Twelfth International Conference on Probabilistic Graphical Models (PGM-2024)*. PMLR, pp. 38–56.

# Buckets

- ▶ Buckets count the occurrences of specific range values in an assignment
- ▶ Each potential belongs to *exactly one* bucket
- ▶ Each bucket contains *at least one* potential

$A$	$B$	$R$	$\phi(A, B, R)$	$b$												
true	true	true	$\varphi_1$	[3, 0]	<table><tr><th><math>b</math></th><th><math>\phi(b)</math></th></tr><tr><td>[3, 0]</td><td><math>\langle \varphi_1 \rangle</math></td></tr><tr><td>[2, 1]</td><td><math>\langle \varphi_4, \varphi_2, \varphi_2 \rangle</math></td></tr><tr><td>[1, 2]</td><td><math>\langle \varphi_5, \varphi_5, \varphi_3 \rangle</math></td></tr><tr><td>[0, 3]</td><td><math>\langle \varphi_6 \rangle</math></td></tr></table>	$b$	$\phi(b)$	[3, 0]	$\langle \varphi_1 \rangle$	[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$	[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$	[0, 3]	$\langle \varphi_6 \rangle$	
$b$	$\phi(b)$															
[3, 0]	$\langle \varphi_1 \rangle$															
[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$															
[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$															
[0, 3]	$\langle \varphi_6 \rangle$															
true	true	false	$\varphi_4$	[2, 1]												
true	false	true	$\varphi_2$	[2, 1]												
true	false	false	$\varphi_5$	[1, 2]												
false	true	true	$\varphi_2$	[2, 1]												
false	true	false	$\varphi_5$	[1, 2]												
false	false	true	$\varphi_3$	[1, 2]												
false	false	false	$\varphi_6$	[0, 3]												

# Properties of Buckets I

For any subset  $\mathcal{S}$  of commutative arguments:

- ▶  $|\mathcal{S}| \leq \min_{b \in \{b | b \in \mathcal{B}(\phi) \wedge |\phi(b)| > 1\}} \max_{\varphi \in \phi(b)} \text{count}(\phi(b), \varphi)$ , i.e.,
- ▶ in each bucket  $b$  with  $|\phi(b)| > 1$ , there is a potential occurring at least  $|\mathcal{S}|$  times

$A$	$B$	$R$	$\phi(A, B, R)$	$b$
true	true	true	$\varphi_1$	[3, 0]
true	true	false	$\varphi_4$	[2, 1]
true	false	true	$\varphi_2$	[2, 1]
true	false	false	$\varphi_5$	[1, 2]
false	true	true	$\varphi_2$	[2, 1]
false	true	false	$\varphi_5$	[1, 2]
false	false	true	$\varphi_3$	[1, 2]
false	false	false	$\varphi_6$	[0, 3]

$b$	$\phi(b)$
[3, 0]	$\langle \varphi_1 \rangle$
[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
[0, 3]	$\langle \varphi_6 \rangle$

## Properties of Buckets II

For a group of identical potentials in a bucket:

- Intersection of their corresponding assignments yields candidates
- E.g., for  $\varphi_2$ 's in  $[2, 1]$ :  $(\text{true}, \text{false}, \text{true}) \cap (\text{false}, \text{true}, \text{true}) = (\emptyset, \emptyset, \text{true})$

$A$	$B$	$R$	$\phi(A, B, R)$	$b$
true	true	true	$\varphi_1$	$[3, 0]$
true	true	false	$\varphi_4$	$[2, 1]$
true	false	true	$\varphi_2$	$[2, 1]$
true	false	false	$\varphi_5$	$[1, 2]$
false	true	true	$\varphi_2$	$[2, 1]$
false	true	false	$\varphi_5$	$[1, 2]$
false	false	true	$\varphi_3$	$[1, 2]$
false	false	false	$\varphi_6$	$[0, 3]$

$b$	$\phi(b)$
$[3, 0]$	$\langle \varphi_1 \rangle$
$[2, 1]$	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
$[1, 2]$	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
$[0, 3]$	$\langle \varphi_6 \rangle$

# The Detection of Commutative Factors Algorithm – Example Run

- Iterate over buckets and compute candidates for commutative arguments

$A$	$B$	$R$	$\phi(A, B, R)$	$b$	$b$	$\phi(b)$
true	true	true	$\varphi_1$	[3, 0]	[3, 0]	$\langle \varphi_1 \rangle$
true	true	false	$\varphi_4$	[2, 1]	[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
true	false	true	$\varphi_2$	[2, 1]	[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
true	false	false	$\varphi_5$	[1, 2]	[0, 3]	$\langle \varphi_6 \rangle$
false	true	true	$\varphi_2$	[2, 1]		
false	true	false	$\varphi_5$	[1, 2]		
false	false	true	$\varphi_3$	[1, 2]		
false	false	false	$\varphi_6$	[0, 3]		

Initial candidates:  $\{\{A, B, R\}\}$

# The Detection of Commutative Factors Algorithm – Example Run

- Iterate over buckets and compute candidates for commutative arguments

$A$	$B$	$R$	$\phi(A, B, R)$	$b$	$b$	$\phi(b)$
true	true	true	$\varphi_1$	[3, 0]	[3, 0]	$\langle \varphi_1 \rangle$
true	true	false	$\varphi_4$	[2, 1]	[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
true	false	true	$\varphi_2$	[2, 1]	[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
true	false	false	$\varphi_5$	[1, 2]	[0, 3]	$\langle \varphi_6 \rangle$
false	true	true	$\varphi_2$	[2, 1]		
false	true	false	$\varphi_5$	[1, 2]		
false	false	true	$\varphi_3$	[1, 2]		
false	false	false	$\varphi_6$	[0, 3]		

Initial candidates:  $\{\{A, B, R\}\}$   
 $[3, 0]$  ( $|\phi(b)| < 2$ ):  $\{\{A, B, R\}\}$

# The Detection of Commutative Factors Algorithm – Example Run

- Iterate over buckets and compute candidates for commutative arguments

$A$	$B$	$R$	$\phi(A, B, R)$	$b$	$b$	$\phi(b)$
true	true	true	$\varphi_1$	[3, 0]	[3, 0]	$\langle \varphi_1 \rangle$
true	true	false	$\varphi_4$	[2, 1]	[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
true	false	true	$\varphi_2$	[2, 1]	[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
true	false	false	$\varphi_5$	[1, 2]	[0, 3]	$\langle \varphi_6 \rangle$
false	true	true	$\varphi_2$	[2, 1]		
false	true	false	$\varphi_5$	[1, 2]		
false	false	true	$\varphi_3$	[1, 2]		
false	false	false	$\varphi_6$	[0, 3]		

Initial candidates:  $\{\{A, B, R\}\}$   
 $[3, 0]$  ( $|\phi(b)| < 2$ ):  $\{\{A, B, R\}\}$   
 $[2, 1]$ :  $\{\{A, B\}\}$

$$(\text{true}, \text{false}, \text{true}) \cap (\text{false}, \text{true}, \text{true}) = (\emptyset, \emptyset, \text{true})$$

# The Detection of Commutative Factors Algorithm – Example Run

- Iterate over buckets and compute candidates for commutative arguments

$A$	$B$	$R$	$\phi(A, B, R)$	$b$	$b$	$\phi(b)$
true	true	true	$\varphi_1$	[3, 0]	[3, 0]	$\langle \varphi_1 \rangle$
true	true	false	$\varphi_4$	[2, 1]	[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
true	false	true	$\varphi_2$	[2, 1]	[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
true	false	false	$\varphi_5$	[1, 2]	[0, 3]	$\langle \varphi_6 \rangle$
false	true	true	$\varphi_2$	[2, 1]		
false	true	false	$\varphi_5$	[1, 2]		
false	false	true	$\varphi_3$	[1, 2]		
false	false	false	$\varphi_6$	[0, 3]		

Initial candidates:  $\{\{A, B, R\}\}$   
 $[3, 0]$  ( $|\phi(b)| < 2$ ):  $\{\{A, B, R\}\}$   
 $[2, 1]$ :  $\{\{A, B\}\}$   
 $[1, 2]$ :  $\{\{A, B\}\}$

$$(true, false, false) \cap (false, true, false) = (\emptyset, \emptyset, false)$$



# The Detection of Commutative Factors Algorithm – Example Run

- Iterate over buckets and compute candidates for commutative arguments

$A$	$B$	$R$	$\phi(A, B, R)$	$b$
true	true	true	$\varphi_1$	[3, 0]
true	true	false	$\varphi_4$	[2, 1]
true	false	true	$\varphi_2$	[2, 1]
true	false	false	$\varphi_5$	[1, 2]
false	true	true	$\varphi_2$	[2, 1]
false	true	false	$\varphi_5$	[1, 2]
false	false	true	$\varphi_3$	[1, 2]
false	false	false	$\varphi_6$	[0, 3]

$b$	$\phi(b)$
[3, 0]	$\langle \varphi_1 \rangle$
[2, 1]	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
[1, 2]	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
[0, 3]	$\langle \varphi_6 \rangle$

Initial candidates:  $\{\{A, B, R\}\}$   
 $[3, 0]$  ( $|\phi(b)| < 2$ ):  $\{\{A, B, R\}\}$   
 $[2, 1]$ :  $\{\{A, B\}\}$   
 $[1, 2]$ :  $\{\{A, B\}\}$   
 $[0, 3]$  ( $|\phi(b)| < 2$ ):  $\{\{A, B\}\}$

# The Detection of Commutative Factors Algorithm – Complexity

- ▶ Avoids »naive« iteration over all  $2^n$  subsets of arguments
- ▶ Time complexity is upper-bounded depending on the number of groups of identical potentials in the buckets

$b$	$\phi(b)$
$[3, 0]$	$\langle \varphi_1 \rangle$
$[2, 1]$	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
$[1, 2]$	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
$[0, 3]$	$\langle \varphi_6 \rangle$

# The Detection of Commutative Factors Algorithm – Complexity

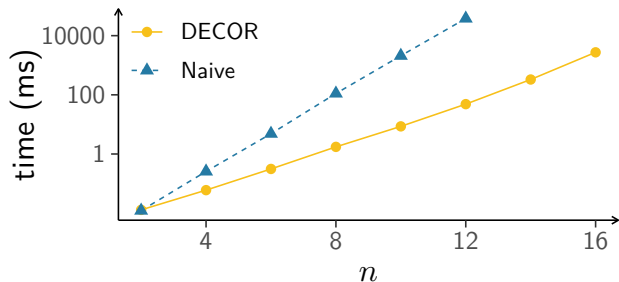
- ▶ Avoids »naive« iteration over all  $2^n$  subsets of arguments
- ▶ Time complexity is upper-bounded depending on the number of groups of identical potentials in the buckets
- ▶ In practice, there are two possible scenarios:

$b$	$\phi(b)$
$[3, 0]$	$\langle \varphi_1 \rangle$
$[2, 1]$	$\langle \varphi_4, \varphi_2, \varphi_2 \rangle$
$[1, 2]$	$\langle \varphi_5, \varphi_5, \varphi_3 \rangle$
$[0, 3]$	$\langle \varphi_6 \rangle$

1. A factor contains commutative arguments
  - ▶ Potential values are likely to contain a single group of duplicates
2. A factor contains no commutative arguments
  - ▶ Potential values are likely to be distinct

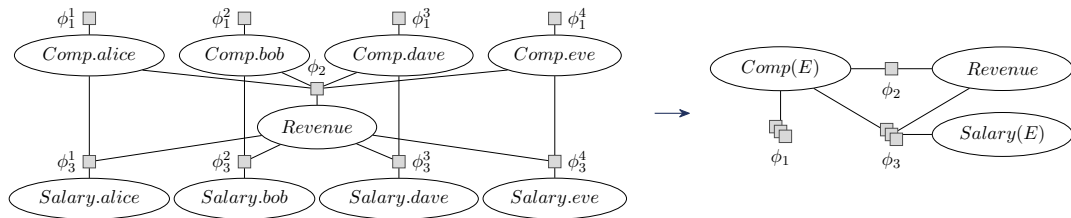
# The Detection of Commutative Factors Algorithm – Experiments

- ▶ Comparison of run times of DECOR and the »naive« approach
- ▶ Average over factors with  $k \in \{0, 2, \lfloor \frac{n}{2} \rfloor, n-1, n\}$  commutative arguments
- ▶ Timeout after five minutes per instance



# The Detection of Commutative Factors Algorithm – Summary

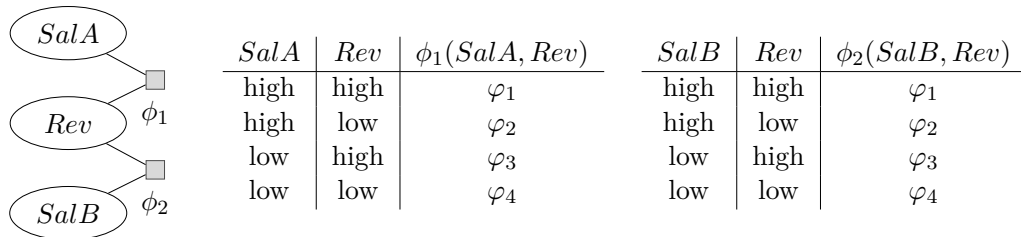
- ▶ Problem of efficiently detecting commutative factors solved
- ▶ Upper bound on the number of iterations depending on the number of groups of identical potential values within the buckets of a factor
- ▶ The DECOR algorithm effectively exploits this upper bound in practice



# Approximate Lifted Model Construction

► So far: Strict equality between potentials required

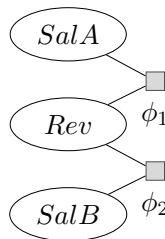
► E.g.,  $\varphi_1 = \varphi_1$ ,  $\varphi_2 = \varphi_2$ ,  $\varphi_3 = \varphi_3$ ,  $\varphi_4 = \varphi_4$



# Approximate Lifted Model Construction

► Next: Allow for a small deviation between potentials for practical applicability

► E.g.,  $\varphi_1 \approx \varphi'_1$ ,  $\varphi_2 \approx \varphi'_2$ ,  $\varphi_3 \approx \varphi'_3$ ,  $\varphi_4 \approx \varphi'_4$



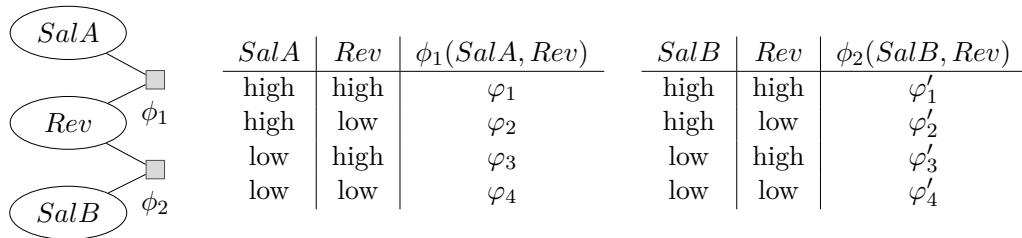
<i>SalA</i>	<i>Rev</i>	$\phi_1(\textit{SalA}, \textit{Rev})$	<i>SalB</i>	<i>Rev</i>	$\phi_2(\textit{SalB}, \textit{Rev})$
high	high	$\varphi_1$	high	high	$\varphi'_1$
high	low	$\varphi_2$	high	low	$\varphi'_2$
low	high	$\varphi_3$	low	high	$\varphi'_3$
low	low	$\varphi_4$	low	low	$\varphi'_4$

Malte Luttermann, Jan Speller, Marcel Gehrke, Tanya Braun, Ralf Möller, and Mattis Hartwig (2025).

»Approximate Lifted Model Construction«. *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-2025)*. IJCAI Organization.

# Approximate Lifted Model Construction

- ▶ Next: Allow for a small deviation between potentials for practical applicability
  - ▶ E.g.,  $\varphi_1 \approx \varphi'_1$ ,  $\varphi_2 \approx \varphi'_2$ ,  $\varphi_3 \approx \varphi'_3$ ,  $\varphi_4 \approx \varphi'_4$



- ▶ How can factors that are not strictly equivalent be grouped?
- ▶ How much deviation should be allowed and what is the impact on query results?



## $\varepsilon$ -Equivalence

- Potentials  $\varphi_1 \in \mathbb{R}^+$  and  $\varphi_2 \in \mathbb{R}^+$  are  $\varepsilon$ -equivalent if

$$\varphi_1 \in [\varphi_2 \cdot (1 - \varepsilon), \varphi_2 \cdot (1 + \varepsilon)] \text{ and}$$

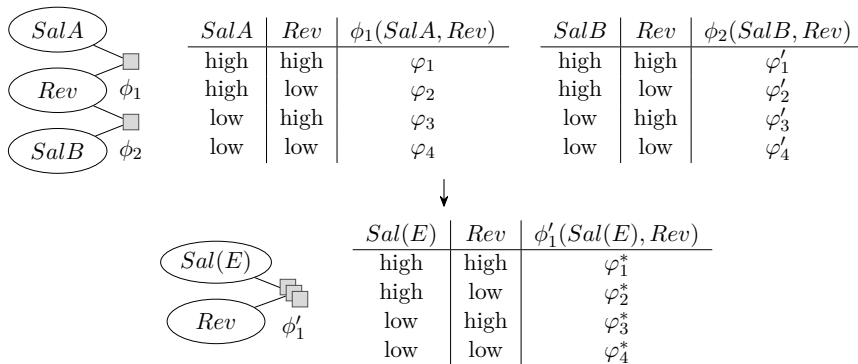
$$\varphi_2 \in [\varphi_1 \cdot (1 - \varepsilon), \varphi_1 \cdot (1 + \varepsilon)]$$

- Factors  $\phi_1(R_1, \dots, R_n)$  and  $\phi_2(R'_1, \dots, R'_n)$  are  $\varepsilon$ -equivalent if all potentials in their potential tables are  $\varepsilon$ -equivalent
- E.g., for  $\varepsilon = 0.1$ ,  $\phi_1(\text{SalA}, \text{Rev})$  and  $\phi_2(\text{SalB}, \text{Rev})$  are  $\varepsilon$ -equivalent:

<i>SalA</i>	<i>Rev</i>	$\phi_1(\text{SalA}, \text{Rev})$	<i>SalB</i>	<i>Rev</i>	$\phi_2(\text{SalB}, \text{Rev})$
high	high	0.81	high	high	0.84
high	low	0.32	high	low	0.31
low	high	0.51	low	high	0.51
low	low	0.21	low	low	0.20

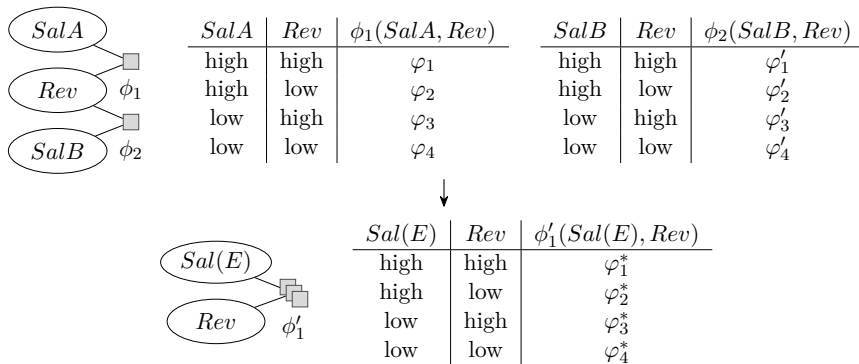
# Grouping $\varepsilon$ -Equivalent Factors

- A representative potential table is required to construct a lifted representation



# Grouping $\varepsilon$ -Equivalent Factors

- A representative potential table is required to construct a lifted representation



- How to choose  $\varphi_i^*$ ?

# Minimisation of the Approximation Error I

For a group of pairwise  $\varepsilon$ -equivalent factors  $\mathbf{G} = \{\phi_1, \dots, \phi_k\}$ , we want to set

$$\phi_1 = \phi^*, \dots, \phi_k = \phi^*$$

# Minimisation of the Approximation Error I

For a group of pairwise  $\varepsilon$ -equivalent factors  $G = \{\phi_1, \dots, \phi_k\}$ , we want to set

$$\phi_1 = \phi^*, \dots, \phi_k = \phi^*$$

such that

$$\phi^* = \arg \min_{\phi_j} \sum_{\phi_i \in G} Err(\phi_i, \phi_j),$$

# Minimisation of the Approximation Error I

For a group of pairwise  $\varepsilon$ -equivalent factors  $G = \{\phi_1, \dots, \phi_k\}$ , we want to set

$$\phi_1 = \phi^*, \dots, \phi_k = \phi^*$$

such that

$$\phi^* = \arg \min_{\phi_j} \sum_{\phi_i \in G} Err(\phi_i, \phi_j),$$

where  $Err(\phi_i, \phi_j)$  is the sum of squared deviations between  $\phi_i$ 's and  $\phi_j$ 's potentials:

$$Err(\phi_i, \phi_j) = \sum_{(r_1, \dots, r_n)} \left( \phi_i(r_1, \dots, r_n) - \phi_j(r_1, \dots, r_n) \right)^2$$

# Minimisation of the Approximation Error II

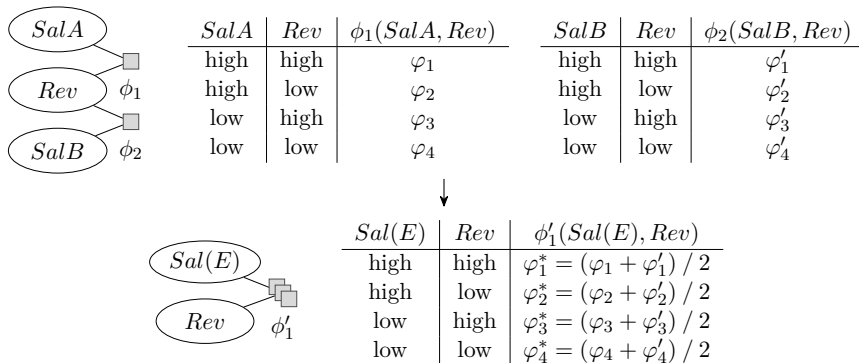
## Theorem

The optimal choice for  $\phi^*$  is the arithmetic mean of the potentials in  $\mathcal{G}$ :

$$\phi^*(r_1, \dots, r_n) = \frac{1}{k} \sum_{i=1}^k \phi_i(r_1, \dots, r_n)$$

# Minimisation of the Approximation Error III

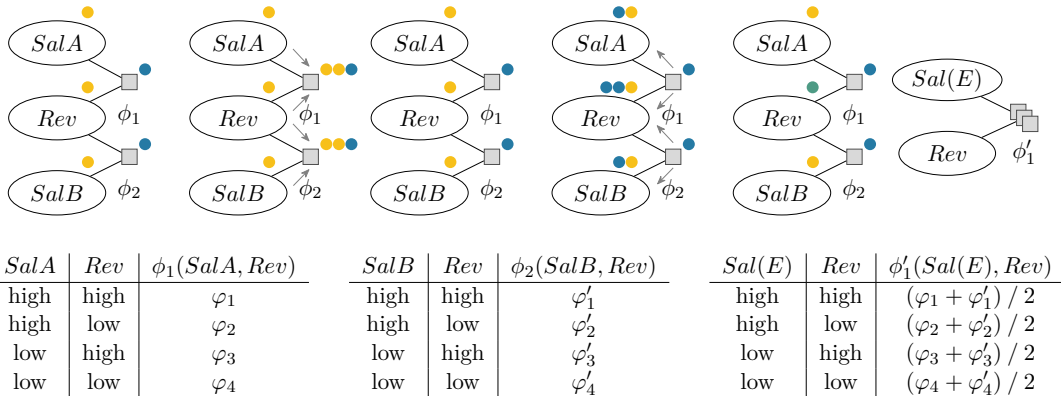
- Choose  $\varphi_i^*$  as the row-wise arithmetic mean of the potentials





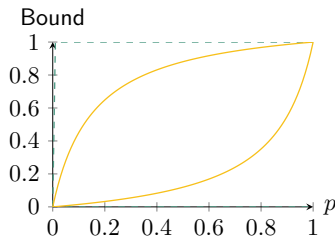
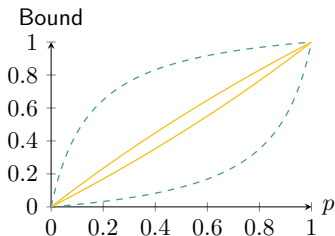
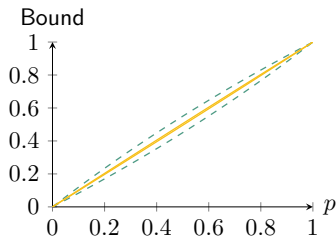
# The $\varepsilon$ -Advanced Colour Passing Algorithm

- Check for  $\varepsilon$ -equivalence instead of strict equivalence between factors



# Bounding the Change in Query Results

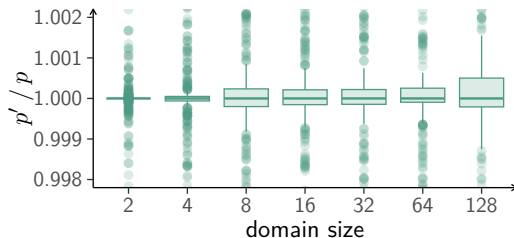
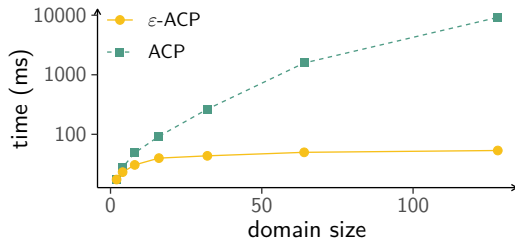
- ▶ Bounds for  $m = 10$  (left),  $m = 100$  (middle), and  $m = 1000$  (right) factors
- ▶ Dashed line:  $\varepsilon = 0.01$ , solid line:  $\varepsilon = 0.001$
- ▶ x-axes depict the original probability  $p$ , y-axes reflect the bound on the change in  $p$



- ▶ Bounds apply to arbitrary queries and factor graphs

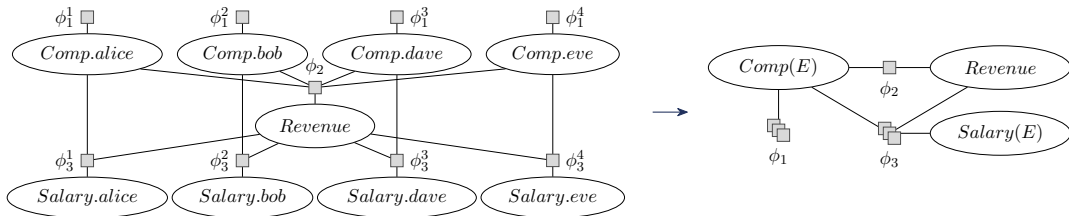
# The $\epsilon$ -Advanced Colour Passing Algorithm – Experiments

- ▶  $\epsilon$ -ACP is a generalisation of ACP that assigns identical colours to groups of  $\epsilon$ -equivalent factors (instead of equivalent factors)
- ▶ Left: Comparison of run times for lifted inference
- ▶ Right: Quotients of query results  $p'$  in the modified factor graph and  $p$  in the original factor graph



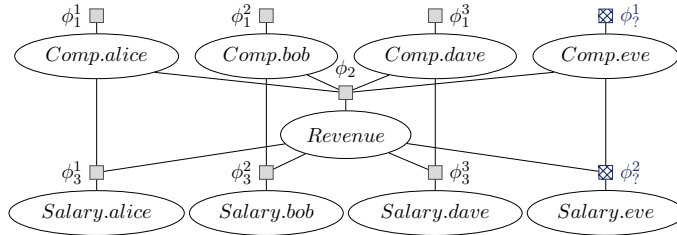
# The $\varepsilon$ -Advanced Colour Passing Algorithm – Summary

- ▶ Limited practical applicability of ACP solved
- ▶ Hyperparameter  $\varepsilon$  to control the trade-off between exactness and compactness
- ▶ Theoretical guarantees for the change in query results



# Lifted Model Construction with Unknown Factors

- How to handle unknown factors (i.e., factors with missing function definitions)?



Malte Luttermann, Ralf Möller, and Marcel Gehrke (2023). »Lifting Factor Graphs with Some Unknown Factors«. *Proceedings of the Seventeenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-2023)*. Springer, pp. 337–347.

Malte Luttermann, Ralf Möller, and Marcel Gehrke (2025). »Lifting Factor Graphs with Some Unknown Factors for New Individuals«. *International Journal of Approximate Reasoning* 179, p. 109371.

# Lifted Model Construction with Unknown Factors – Overview

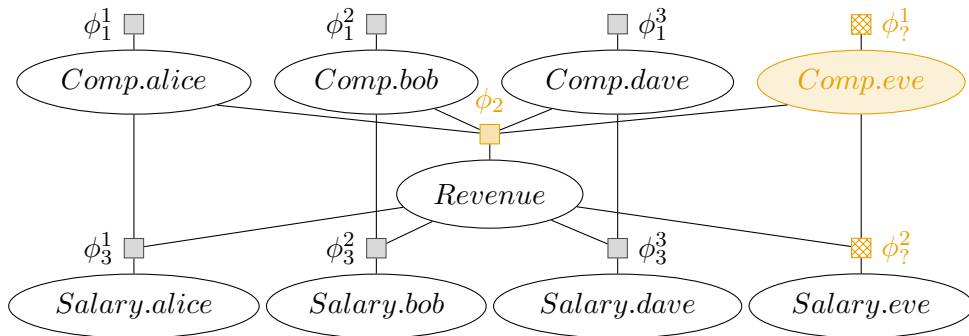
- ▶ What colour to assign to the unknown factors?
  - ▶ Potentials are missing
  - ▶ Only available information: Surrounding graph structure

# Lifted Model Construction with Unknown Factors – Overview

- ▶ What colour to assign to the unknown factors?
  - ▶ Potentials are missing
  - ▶ Only available information: Surrounding graph structure
- ▶ General idea:
  1. Known factors are coloured according to their potentials
  2. Unknown factors are coloured according to their 2-step neighbourhood
  3. Assign unknown factors and known factors the same colour if their 2-step neighbourhoods are symmetric
  4. Run the standard ACP algorithm using the previously assigned colours

## 2-Step Neighbourhood

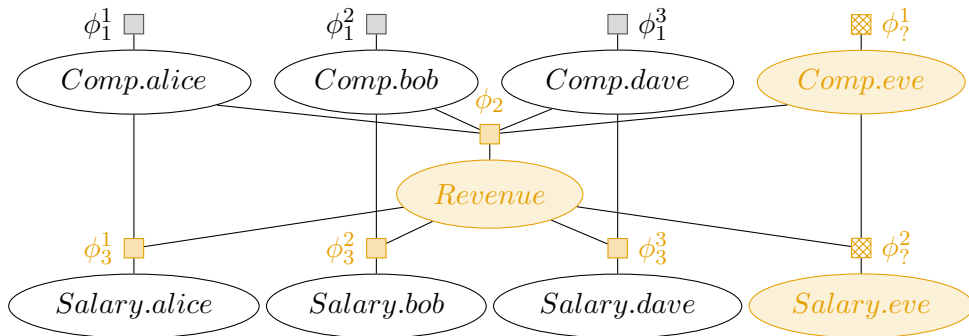
- ▶ The 2-step neighbourhood of a factor  $\phi$  contains all variable nodes directly connected to  $\phi$  and all factors that are direct neighbours of any variable node connected to  $\phi$
- ▶ E.g.,  $2\text{-step}(\phi_7^1) = \{Comp.eve\} \cup \{\phi_2, \phi_7^1, \phi_7^2\}$





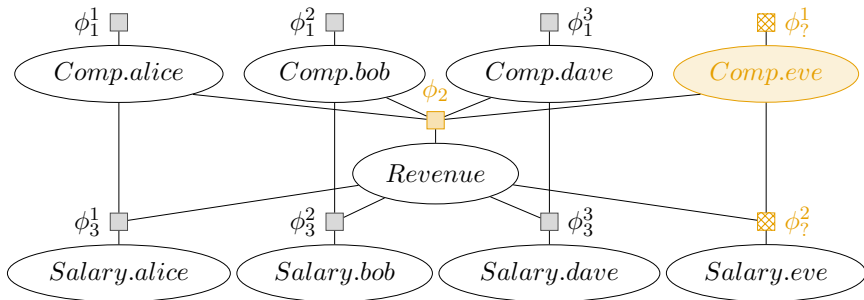
## 2-Step Neighbourhood

- ▶ The 2-step neighbourhood of a factor  $\phi$  contains all variable nodes directly connected to  $\phi$  and all factors that are direct neighbours of any variable node connected to  $\phi$
- ▶ E.g.,  $2\text{-step}(\phi_7^2) = \{Comp.eve, Salary.eve, Revenue\} \cup \{\phi_3^1, \phi_3^2, \phi_3^3, \phi_2, \phi_7^1, \phi_7^2\}$



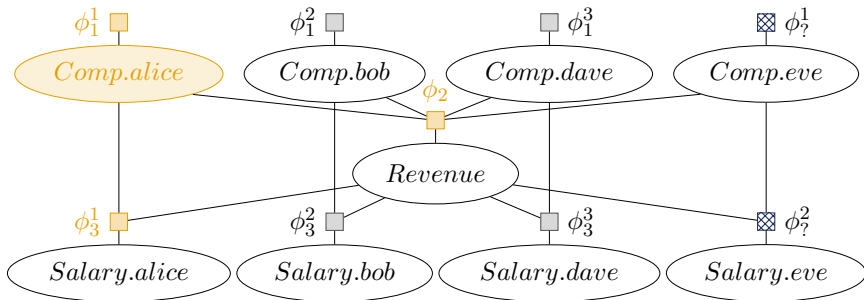
# Symmetric 2-Step Neighbourhoods

- ▶  $\phi_i$  and  $\phi_j$  have symmetric 2-step neighbourhoods if  $\phi_i$  and  $\phi_j$  have the same number of neighbours and there is a one-to-one correspondence of the neighbours of  $\phi_i$  and  $\phi_j$  such that their observed events, ranges, and numbers of neighbours are identical
- ▶ E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_2^2)$ , and  $2\text{-step}(\phi_3^3)$



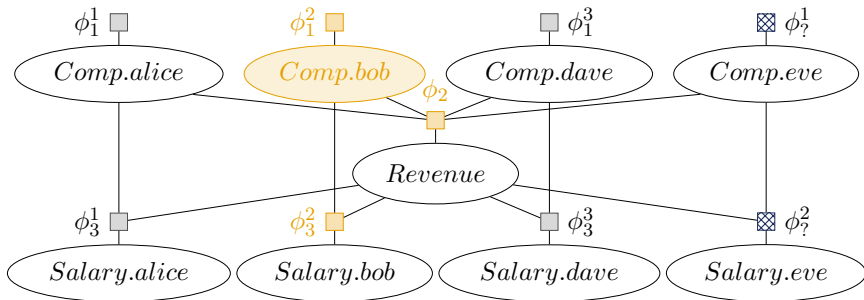
# Symmetric 2-Step Neighbourhoods

- ▶  $\phi_i$  and  $\phi_j$  have symmetric 2-step neighbourhoods if  $\phi_i$  and  $\phi_j$  have the same number of neighbours and there is a one-to-one correspondence of the neighbours of  $\phi_i$  and  $\phi_j$  such that their observed events, ranges, and numbers of neighbours are identical
- ▶ E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_2^2)$ , and  $2\text{-step}(\phi_3^3)$



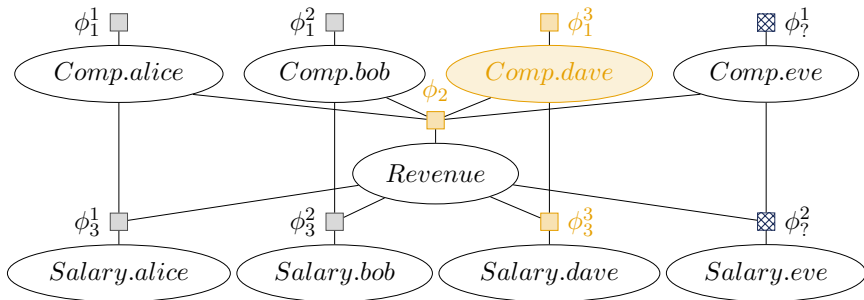
## Symmetric 2-Step Neighbourhoods

- ▶  $\phi_i$  and  $\phi_j$  have symmetric 2-step neighbourhoods if  $\phi_i$  and  $\phi_j$  have the same number of neighbours and there is a one-to-one correspondence of the neighbours of  $\phi_i$  and  $\phi_j$  such that their observed events, ranges, and numbers of neighbours are identical
- ▶ E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_2^2)$ , and  $2\text{-step}(\phi_3^3)$



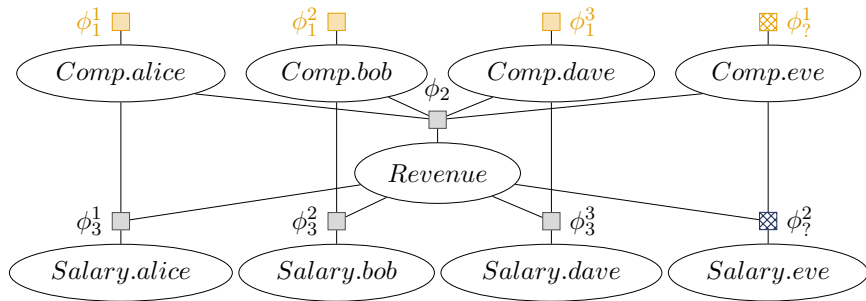
# Symmetric 2-Step Neighbourhoods

- ▶  $\phi_i$  and  $\phi_j$  have symmetric 2-step neighbourhoods if  $\phi_i$  and  $\phi_j$  have the same number of neighbours and there is a one-to-one correspondence of the neighbours of  $\phi_i$  and  $\phi_j$  such that their observed events, ranges, and numbers of neighbours are identical
- ▶ E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_2^2)$ , and  $2\text{-step}(\phi_3^3)$



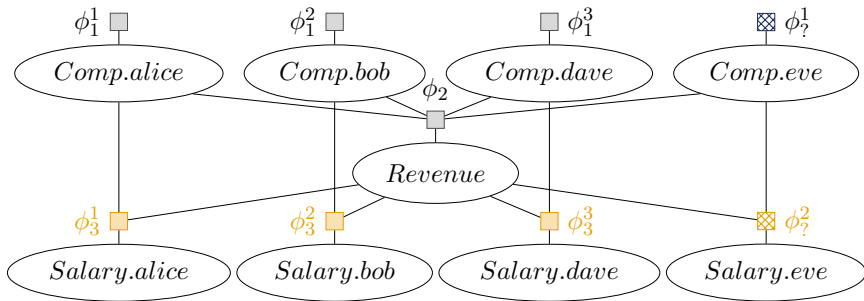
# Transfer of Potentials

- Potentials from known factors can be transferred to unknown factors
  - E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_1^2)$ , and  $2\text{-step}(\phi_1^3)$
  - E.g.,  $2\text{-step}(\phi_7^2)$  is symmetric to  $2\text{-step}(\phi_3^1)$ ,  $2\text{-step}(\phi_3^2)$ , and  $2\text{-step}(\phi_3^3)$



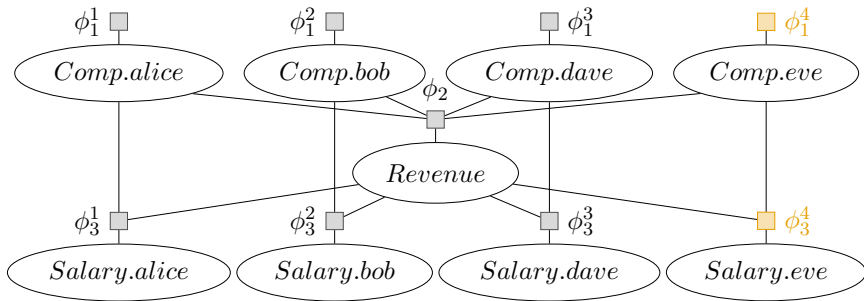
# Transfer of Potentials

- Potentials from known factors can be transferred to unknown factors
  - E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_1^2)$ , and  $2\text{-step}(\phi_1^3)$
  - E.g.,  $2\text{-step}(\phi_7^2)$  is symmetric to  $2\text{-step}(\phi_3^1)$ ,  $2\text{-step}(\phi_3^2)$ , and  $2\text{-step}(\phi_3^3)$



# Transfer of Potentials

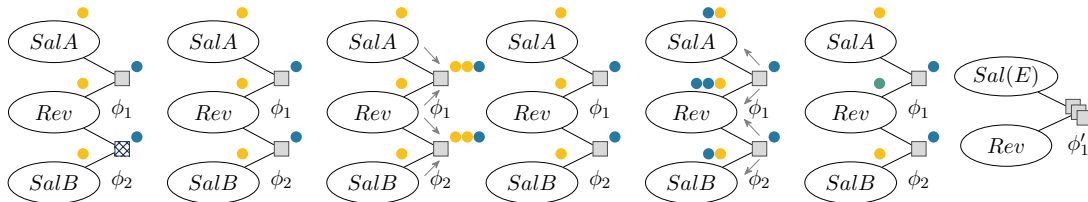
- ▶ Potentials from known factors can be transferred to unknown factors
  - ▶ E.g.,  $2\text{-step}(\phi_7^1)$  is symmetric to  $2\text{-step}(\phi_1^1)$ ,  $2\text{-step}(\phi_1^2)$ , and  $2\text{-step}(\phi_1^3)$
  - ▶ E.g.,  $2\text{-step}(\phi_7^2)$  is symmetric to  $2\text{-step}(\phi_3^1)$ ,  $2\text{-step}(\phi_3^2)$ , and  $2\text{-step}(\phi_3^3)$





# The Lifting Factor Graphs with Some Unknown Factors Algorithm

- 2-step neighbourhoods to assign colours (and possibly potentials) to unknown factors



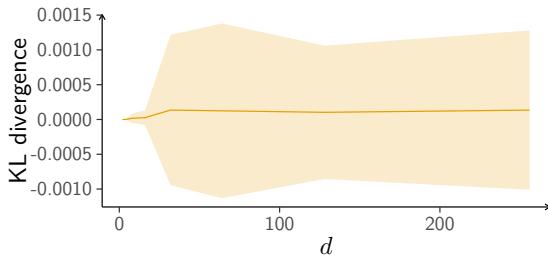
<i>SalA</i>	<i>Rev</i>	$\phi_1(\text{SalA}, \text{Rev})$
high	high	$\varphi_1$
high	low	$\varphi_2$
low	high	$\varphi_3$
low	low	$\varphi_4$

<i>SalB</i>	<i>Rev</i>	$\phi_2(\text{SalB}, \text{Rev})$
high	high	$? \rightarrow \varphi_1$
high	low	$? \rightarrow \varphi_2$
low	high	$? \rightarrow \varphi_3$
low	low	$? \rightarrow \varphi_4$

<i>Sal(E)</i>	<i>Rev</i>	$\phi'_1(\text{Sal(E)}, \text{Rev})$
high	high	$\varphi_1$
high	low	$\varphi_2$
low	high	$\varphi_3$
low	low	$\varphi_4$

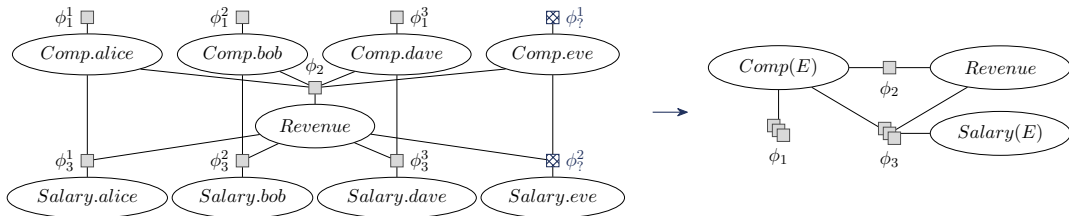
## Lifting Factor Graphs with Some Unknown Factors – Experiments

- ▶ Generate factor graphs where all factors are known
- ▶ Randomly remove potentials of 5-10 percent of the factors
- ▶ Run the Lifting Factor Graphs with Some Unknown Factors algorithm to obtain a lifted representation
- ▶ Perform probabilistic inference on the ground truth and the lifted representation



# Lifting Factor Graphs with Some Unknown Factors – Summary

- ▶ Lifted model construction with unknown factors solved
- ▶ Transfer of known potentials to unknown factors
- ▶ Establishment of a well-defined semantics if possible






# Agenda

1. Introduction to relational models [Marcel]
2. Compressing probabilistic relational models [Malte]
3. Application: Lifted causal inference [Malte]
  - ▶ Lifted computation of causal effects
  - ▶ Lifted computation of causal effects with partial causal knowledge
4. Summary [Marcel]



# Bibliography I

-  Babak Ahmadi, Kristian Kersting, Martin Mladenov, and Sriraam Natarajan (2013). »Exploiting Symmetries for Scaling Loopy Belief Propagation and Relational Training«. *Machine Learning* 92, pp. 91–132.
-  Kristian Kersting, Babak Ahmadi, and Sriraam Natarajan (2009). »Counting Belief Propagation«. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI-2009)*. AUAI Press, pp. 277–284.
-  Malte Luttermann, Tanya Braun, Ralf Möller, and Marcel Gehrke (2024). »Colour Passing Revisited: Lifted Model Construction with Commutative Factors«. *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-2024)*. AAAI Press, pp. 20500–20507.

## Bibliography II

-  Malte Luttermann, Johann Machemer, and Marcel Gehrke (2024a). »Efficient Detection of Commutative Factors in Factor Graphs«. *Proceedings of the Twelfth International Conference on Probabilistic Graphical Models (PGM-2024)*. PMLR, pp. 38–56.
-  — (2024b). »Efficient Detection of Exchangeable Factors in Factor Graphs«. *Proceedings of the Thirty-Seventh International Florida Artificial Intelligence Research Society Conference (FLAIRS-2024)*. **Best Student Paper**. Florida Online Journals.
-  Malte Luttermann, Ralf Möller, and Marcel Gehrke (2023). »Lifting Factor Graphs with Some Unknown Factors«. *Proceedings of the Seventeenth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU-2023)*. Springer, pp. 337–347.

## Bibliography III

-  Malte Luttermann, Ralf Möller, and Marcel Gehrke (2025). »Lifting Factor Graphs with Some Unknown Factors for New Individuals«. *International Journal of Approximate Reasoning* 179, p. 109371.
-  Malte Luttermann, Jan Speller, Marcel Gehrke, Tanya Braun, Ralf Möller, and Mattis Hartwig (2025). »Approximate Lifted Model Construction«. *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence (IJCAI-2025)*. IJCAI Organization.