Faster Lifting for Two-Variable Logic Using Cell Graphs (Supplementary Material)

Timothy van Bremen[1] **Ondřej Kuželka**[2]

[1]KU Leuven, Belgium
[2]Czech Technical University in Prague, Czech Republic

1 COMPLETE PSEUDOCODE

The pseudocode for the functions GetGTerm and GetJTerm are given in Algorithms 1 and 2. The latter also makes use of another function, GetDTerm, which is given in Algorithm 3.

Algorithm 1 GetGTerm

Input: p, N, $n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|}$
Output: $g_p(n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|}, N)$

if $[p, N, n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|}]$ in cache **then**
    return cache$[p, N, n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|}]$
$s \leftarrow 0$
if $p = 0$ **then**
    for $i \in \{1, \ldots, k\}$ **do**
        $t \leftarrow w_i$
        for $j \in \{k+l+1, \ldots, |\mathcal{F}_C|\}$ **do**
            $t \leftarrow r_{ij}^{n_j}$
        $s \leftarrow s + t$
    $s \leftarrow s^{n-N-n_{k+l+1}-\cdots-n_{|\mathcal{F}_C|}}$
else
    for $n_{k+p} \in \{0, \ldots, n-N-n_{k+l+1}-\cdots-n_{|\mathcal{F}_C|}\}$ **do**
        $t \leftarrow \binom{n-N-n_{k+l+1}-\cdots-n_{|\mathcal{F}_C|}}{n_{k+p}} \cdot w_{k+p}^{n_{k+p}}$
        $t \leftarrow t \cdot$ GetJTerm$(k+p, n_{k+p})$
        for $j \in \{k+l+1, \ldots, |\mathcal{F}_C|\}$ **do**
            $t \leftarrow t \cdot r_{pj}^{n_p n_j} \cdot$ GetGTerm$(p-1, N+n_{k+p}, n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|})$
        $s \leftarrow s + t$
cache$[p, N, n_{k+l+1}, \ldots, n_{|\mathcal{F}_C|}] \leftarrow s$
return s

Algorithm 2 GetJTerm

Input: c, $\widehat{n}$
Output: $J_c(\widehat{n})$

if $[c, \widehat{n}]$ in cache **then**
    return $[c, \widehat{n}]$
/ s and r are arbitrary r_{ij} and s_i terms in c */*:
if c is a clique of length 1 **then**
    $o \leftarrow s^{\widehat{n}(\widehat{n}-1)/2}$
else
    $o \leftarrow r^{\widehat{n}(\widehat{n}-1)/2} \cdot$ GetDTerm$(c, 1, \widehat{n},$ CliqueLength$(c))$
cache$[i, \widehat{n}] \leftarrow o$
return o

Algorithm 3 GetDTerm

Input: c, i, k, $\widehat{n}$
Output: $d_{i,c}(\widehat{n})$

if $[c, i, \widehat{n}]$ in cache **then**
    return $[c, i, \widehat{n}]$
/ s and r are arbitrary r_{ij} and s_i terms in c */*:
if $i = k$ **then**
    $o \leftarrow \left(\frac{s}{r}\right)^{\widehat{n}(\widehat{n}-1)/2}$
else
    $o \leftarrow 0$
    for $n_i \in \{0, \ldots, \widehat{n}\}$ **do**
        $m \leftarrow \binom{\widehat{n}}{n_i} \cdot \left(\frac{s}{r}\right)^{n_i(n_i-1)/2}$
        $m \leftarrow m \cdot$ GetDTerm$(c, i+1, k, \widehat{n} - n_i)$
        $o \leftarrow o + m$
cache$[c, i, \widehat{n}] \leftarrow o$
return o

2 SENTENCES FROM EXPERIMENTS

The first-order logic encodings of the experiment benchmarks are given below. As mentioned in the main text, the `3-regular` and `derangements` benchmarks are specified in $\mathbf{C}^2$, and have been translated into $\mathbf{FO}^2$ sentences (also given below). Support for *cardinality constraints* are required to obtain the final solutions in these instances. The details for enforcing these cardinality constraints are beyond the scope of this paper (one way is to make repeated calls to an $\mathbf{FO}^2$ WFOMC oracle with varying weights), but we refer interested readers to [Kuzelka, 2021] for details.

- `3-regular`:

$$\forall x\, \neg E(x,x)$$
$$\forall x \forall y\, E(x,y) \rightarrow E(y,x)$$
$$\forall x \exists^{=3} y\, E(x,y)$$

  Transformed version:

$$\begin{aligned}
&\forall x\, \neg E(x,x)\\
&\forall x \forall y\, E(x,y) \rightarrow E(y,x)\\
&\forall x \forall y\, E(x,y) \leftrightarrow F(x,y)\\
&\forall x \forall y\, F_1(x,y) \rightarrow S_1(x)\\
&\forall x \forall y\, F_2(x,y) \rightarrow S_2(x)\\
&\forall x \forall y\, F_3(x,y) \rightarrow S_3(x)\\
&\forall x \forall y\, F(x,y) \leftrightarrow F_1(x,y) \vee F_2(x,y) \vee F_3(x,y)\\
&\forall x \forall y\, \neg F_1(x,y) \vee \neg F_2(x,y)\\
&\forall x \forall y\, \neg F_1(x,y) \vee \neg F_3(x,y)\\
&\forall x \forall y\, \neg F_2(x,y) \vee \neg F_3(x,y)
\end{aligned}$$

  with $\bar{w}(S_1) = \bar{w}(S_2) = \bar{w}(S_3) = -1$, and all other weights set to 1. The cardinality constraint one needs to add to obtain the number of 3-regular graphs is then $|F| = 3n$. Since these cardinality constraints are handled in [Kuzelka, 2021] by multiple calls to an oracle for the $\mathbf{FO}^2$ sentence above, we ignore them here as they are not important for comparing the performance of the $\mathbf{FO}^2$ algorithms.

- `4-coloured`:

$$\begin{aligned}
&\forall x\, \neg E(x,x)\\
&\forall x \forall y\, E(x,y) \rightarrow E(y,x)\\
&\forall x\, C_1(x) \vee C_2(x) \vee C_3(x) \vee C_4(x)\\
&\forall x\, \neg C_1(x) \vee \neg C_2(x)\\
&\quad \vdots\\
&\forall x\, \neg C_2(x) \vee \neg C_4(x)\\
&\forall x\, \neg C_3(x) \vee \neg C_4(x)\\
&\forall x \forall y\, E(x,y) \rightarrow ((C_1(x) \wedge \neg C_1(y)) \vee \cdots \vee\\
&\qquad (C_4(x) \wedge \neg C_4(y)))
\end{aligned}$$

  In this case no cardinality constraints are needed.

- `derangements`:

$$\forall x\, \neg F(x,x)$$
$$\forall x \exists^{=1} y F(x,y)$$
$$\forall x \exists^{=1} y F(y,x)$$

  Transformed version:

$$\begin{aligned}
&\forall x\, \neg F(x,x)\\
&\forall x \forall y S_1(x) \vee \neg F(x,y)\\
&\forall x \forall y S_2(x) \vee \neg F(y,x)
\end{aligned}$$

  with $\bar{w}(S_1) = \bar{w}(S_2) = -1$, and all other weights set to 1. The cardinality constraint one would need to add to obtain the number of derangements is $|F| = n$.

- `3-matchings`:

$$\begin{aligned}
&\forall x\, \neg E_1(x,x)\\
&\forall x\, \neg E_2(x,x)\\
&\forall x\, \neg E_3(x,x)\\
&\forall x \forall y\, E_1(x,y) \rightarrow E_1(y,x)\\
&\forall x \forall y\, E_2(x,y) \rightarrow E_2(y,x)\\
&\forall x \forall y\, E_3(x,y) \rightarrow E_3(y,x)\\
&\forall x \exists^{=1} y\, E_1(x,y)\\
&\forall x \exists^{=1} y\, E_2(x,y)\\
&\forall x \exists^{=1} y\, E_3(x,y)\\
&\forall x \forall y\, E_1(x,y) \rightarrow \neg E_2(y,x)\\
&\forall x \forall y\, E_1(x,y) \rightarrow \neg E_3(y,x)\\
&\forall x \forall y\, E_2(x,y) \rightarrow \neg E_3(y,x)
\end{aligned}$$

  Transformed version:

$$\begin{aligned}
&\forall x\, \neg E_1(x,x)\\
&\forall x\, \neg E_2(x,x)\\
&\forall x\, \neg E_3(x,x)\\
&\forall x \forall y\, E_1(x,y) \rightarrow E_1(y,x)\\
&\forall x \forall y\, E_2(x,y) \rightarrow E_2(y,x)\\
&\forall x \forall y\, E_3(x,y) \rightarrow E_3(y,x)\\
&\forall x \forall y\, S_1(x) \vee \neg E_1(x,y)\\
&\forall x \forall y\, S_2(x) \vee \neg E_2(x,y)\\
&\forall x \forall y\, S_3(x) \vee \neg E_3(x,y)\\
&\forall x \forall y\, E_1(x,y) \rightarrow \neg E_2(y,x)\\
&\forall x \forall y\, E_1(x,y) \rightarrow \neg E_3(y,x)\\
&\forall x \forall y\, E_2(x,y) \rightarrow \neg E_3(y,x)
\end{aligned}$$

  with $\bar{w}(S_1) = \bar{w}(S_2) = \bar{w}(S_3) = -1$, and all other weights set to 1. The cardinality constraints one needs to add to obtain the number of ways of constructing three non-overlapping maximal matchings on K_{2n} are $|E_1| = |E_2| = |E_3| = n$

3 PROOF OF THEOREM 2

We have:

$$\mathsf{WFOMC}(\phi, n, w, \bar{w}) = \sum_{n_{k+1}+\cdots+n_{|M|}\leq n} \binom{n}{n_{k+1},\ldots,n_{|M|}} \prod_{i,j:i,j\not\in\{1,2,\ldots,k\} i<j} r_{ij}^{n_i n_j} \prod_{i\not\in\{1,2,\ldots,k\}} w_i^{n_i} s_i^{n_i(n_i-1)/2} \cdot$$

$$\cdot \sum_{n_1+\cdots+n_k=n-n_{k+1}-\ldots n_{|M|}} \binom{n-n_{k+1}-\ldots n_{|M|}}{n_1,n_2,\ldots,n_k} \prod_{j\in\{1,2,\ldots,k\}} w_j^{n_j} \prod_{i\in\{k+1,\ldots,|M|\}} r_{j,i}^{n_j n_i} =$$

$$\sum_{n_{k+1}+\cdots+n_{|M|}\leq n} \binom{n}{n_{k+1},\ldots,n_{|M|}} \prod_{i,j:i,j\not\in\{1,2,\ldots,k\} i<j} r_{ij}^{n_i n_j} \prod_{i\not\in\{1,2,\ldots,k\}} w_i^{n_i} s_i^{n_i(n_i-1)/2} \cdot$$

$$\cdot \sum_{n_1+\cdots+n_k=n-n_{k+1}-\ldots n_{|M|}} \binom{n-n_{k+1}-\ldots n_{|M|}}{n_1,n_2,\ldots,n_k} \prod_{j\in\{1,2,\ldots,k\}} \left(w_j \prod_{i\in\{k+1,\ldots,|M|\}} r_{j,i}^{n_i} \right)^{n_j} =$$

$$\sum_{n_{k+1}+\cdots+n_{|M|}\leq n} \binom{n}{n_{k+1},\ldots,n_{|M|}} \prod_{i,j:i,j\not\in\{1,2,\ldots,k\} i<j} r_{ij}^{n_i n_j} \prod_{i\not\in\{1,2,\ldots,k\}} w_i^{n_i} s_i^{n_i(n_i-1)/2} \cdot \left(\sum_{i=1}^{k} w_i \prod_{j\not\in\{1,2,\ldots,k\}} r_{i,j}^{n_j} \right)^{n-n_{k+1}-\cdots-n_{|M|}}$$

In the derivations above, we used the fact that $s_i = 1$ and $r_{ij} = 1$ for all $i, j \in \{1, 2, \ldots, k\}$ and then we applied the multinomial theorem. This finishes the proof.

References

Ondrej Kuzelka. Weighted first-order model counting in the two-variable fragment with counting quantifiers. *J. Artif. Intell. Res.*, 70:1281–1307, 2021.