A COMPUTING POLYTOPES FOR 2-VARIABLE FORMULAS

In this section we describe an algorithm for constructing relational marginal polytopes given by sets of first-order formulas, each with at most 2 logical variables.

The algorithm described in this section is largely inspired by the WFOMC algorithm from [1]. In what follows in this section, we will denote by $\Omega$ a set of possible worlds over domain $\Delta = \{c_1, \ldots, c_{|\Delta|}\}$ which satisfy a given set $\Phi$ of universally quantified first-order logic sentences.\(^1\)

We need an algorithm which can compute the set $\mathcal{K}(\Phi, \Omega_{\Phi_0})$ defined in Section 5. Let $U$ be the set of all unary predicates in the considered first-order language $L$ and $B$ be the set of all binary predicates (for 2-variable formulas, we may assume w.l.o.g.\(^2\) that $L$ does not contain any literals of arity higher than 2).

In the following, we will use the notion of cells, which was also used in [1]. Given a possible world $\omega$, we say that two constants $c, c' \in \Delta$ are in the same cell if for all $u \in U$ we have $\omega \models u(c)$ iff $\omega \models u(c')$; each cell can then be identified by a subset of $U$ naturally.

**Remark 1.** Suppose that $B = \emptyset$ (i.e. we only have unary predicates) and that $\Phi_0$ and $\Phi$ are constant-free. Then we can construct the set $\mathcal{K}(\Phi, \Omega_{\Phi_0})$ in polynomial time as follows. First, we construct an auxiliary set of all integer partitions of $|\Delta|$:

$$\mathcal{J} = \left\{ (j_1, \ldots, j_{|\Delta|}) \left| \sum_{k=1}^{|\Delta|} j_k = |\Delta| \land \forall k : j_k \geq 0 \right. \right\}$$

The intention is that the $i$-th entry of a vector $J \in \mathcal{J}$ should represent the number of constants $c \in \Delta$ that are in the $i$-th cell (here the cells will be ordered arbitrarily in some order). We can then use the set $\mathcal{J}$ to define a set of possible worlds $\Omega_R \subseteq \Omega_{\Phi_0}$ which will be representative of all the possible worlds in the sense that $\mathcal{K}(\Phi, \Omega_{\Phi_0}) = \{(Q_\omega(\alpha_1), \ldots, Q_\omega(\alpha_l)) | \omega \in \Omega_R\}$. We define the set $\Omega_R$ as follows. First we order (arbitrarily) the constants in $\Delta$ and we do the same with the sets in $2^{|\Delta|}$; we denote by $c_i$ the $i$-th constant and similarly, by $U_i$, the $i$-th subset of $U$. For every $J = (j_1, \ldots, j_{|\Delta|}) \in \mathcal{J}$ we construct:

$$\omega_J = \bigcup_{i=1}^{j_1} \bigcup_{R \in U_1} \{R(c_1)\} \cup \bigcup_{i=j_1+1}^{j_1+j_2} \bigcup_{R \in U_2} \{R(c_1)\} \cup \ldots$$

$$\ldots \cup \bigcup_{i=j_1+\ldots+j_{|\Delta|}}^{j_1+\ldots+j_{|\Delta|}+1} \bigcup_{R \in U_{|\Delta|}} \{R(c_1)\}$$

Then we define $\Omega_R = \{\omega_J | J \in \mathcal{J}\}$. Notice that $|\Omega_R|$ is polynomial in $|\Delta|$. Finally, it is easy to show that we can do the following in polynomial time (i.e. polynomial in $|\Delta|$): (i) to filter out possible worlds that do not satisfy $\Phi_0$ and (ii) to compute $(Q_\omega(\alpha_1), \ldots, Q_\omega(\alpha_l))$.

In the next example we illustrate the construction from the above remark.

**Example 2.** Let $U = \{sm/1\}$ and $\Delta = \{Alice, Bob\}$. Then $\mathcal{J} = \{(0, 2), (1, 1), (2, 0)\}$. Now, for every $J \in \mathcal{J}$, we need to construct the respective $\omega_J$. That is, for the ordering of constants Alice $\prec$ Bob and the ordering of cells $\emptyset \prec \{sm/1\}$, we have:

$$\omega_{(0,2)} = \{sm(Alice), sm(Bob)\},$$

$$\omega_{(1,1)} = \{sm(Bob)\},$$

$$\omega_{(2,0)} = \emptyset.$$
same procedure will need to be repeated for all possible worlds from $\Omega_R$.

**Remark 3.** First, we consider literals of the form $R(c,c)$ where $R \in B$ and $c \in \Delta$. We can notice that these literals can be added already in the construction of $\Omega_R$ (using auxiliary unary predicates), so we will not consider this type of literals here further.

The next remark will provide us with a simple way to construct the set of representatives.

**Remark 4.** Let us suppose that the possible world $\omega_J$, where $J = (j_1,\ldots,j_{|J|}) \in 2^J$, is as in Remark 1. We first discuss how we could generate all possible worlds that could be obtained from $\omega_J$. Let $\Delta_q = \{c\sum_{k=1}^{j_k+1} - c\sum_{k=1}^{j_k}\}$, and $\Delta_r = \{c\sum_{k=1}^{j_k+1} - c\sum_{k=1}^{j_k}\}$. Next we could assign a subset of binary predicates $B$ to each element of the set $\{(c,c') \in (\Delta_q \times \Delta_r) | c \neq c'\}$ (note that the condition $c \neq c'$ is only relevant for $r = q$ and note that we have already taken care of literals of the form $R(c,c)$).

If for instance, $(c_1, c_2)$ got assigned the predicates friends, teammates then we would include the literals friends$(c_1,c_2)$ and teammates$(c_1,c_2)$ to the constructed possible world, and analogically for all the other tuples.

Finally, let us define $\#(B,q,r)$ to be the number of pairs of domain elements from $\Delta_q \times \Delta_r$ which are assigned the subset of binary predicates $B \in 2^B$. We may notice that $Q_\omega(\alpha)$ for any 2-variable quantifier-free formula $\alpha$ will only depend on the numbers $\#_\omega(B,q,r)$ but not on any other details of the possible worlds. The same also holds for the 2-variable universally quantified formulas in $\Phi_0$. Hence, we can construct only representatives with distinct $\#_\omega(B,q,r)$’s using a straightforward generalization of the procedure from Remark 1.

Finally, we need to show that the number of representatives in the set constructed according to Remark 4 has size polynomial in $|\Delta|$. Using Remarks 1, 3 and 4, we can obtain the rather crude upper bound:

$$|\Omega_B^R| \leq (|\Delta| + 1)^{2^{[|\Delta|]}} \cdot (|\Delta| + 1)^{2^{[|\Delta|]}} \cdot 4^{[|\Delta|]}.2^{[|\Delta|]}.$$

Here, the first part comes from Remarks 1 and 3 and the second part from Remark 4. Importantly, the bound is polynomial in $|\Delta|$. Since our main aim in this paper is establishing existence of polynomial-time algorithms for weight learning, we will not try to optimize this bound. In practice, one could probably find the vertices defining the polytope faster using a generic SAT solver as an oracle inside a heuristic algorithm iteratively traversing vertices of the polytope, but that would not lead to an algorithm with runtime polynomial in the size of the domain.

**References**