1.6 Consistency Algorithm

Consistency Algorithm

Exercise 1.

- Clearly (2) $\Rightarrow$ (3) since it always true that

$$\pi_{\bar{x}_i}(S_1 \bowtie \ldots \bowtie S_n) \subseteq S_i \ltimes S_j \subseteq S_i$$

- If (1) then an homomorphism h witnessing $q \to D$ gives suitable $S_i = \{h(\bar{u}_i)\}$, showing (1) $\Rightarrow$ (2).
- Suppose (2), and pick $\bar{u} \in S_1 \bowtie \ldots \bowtie S_n$, which is non empty because the S_i are non empty. We can deduce from $\bar{u}$ a map $h_{\bar{u}}$: it maps variable x to the single element in the projection $\pi_{(x)}(\bar{u})$. Then clearly $h_{\bar{u}}(\bar{x}_i) \in S_i \subset q_i(D)$ and hence $h_{\bar{u}}$ is a homomorphism from q_i to D for every i. We get finally that h is a homomorphism witnessing $q \to D$, which shows (2) $\Rightarrow$ (1).
- Suppose finally (3) and let us show (2). Choose $s_i \in S_i$, and deduce homomorphisms h_i witnessing $q_i \to D$. We can also pick a join tree T rooted at X_i. Our goal here is to show that $s_i \in \pi_{\bar{x}_i}(S_1 \bowtie \ldots \bowtie S_n)$, and we will show that we can choose suitable h_j for all j that produce consistency.
  - First, if $X_k = X_j$, and h_k is already chosen, we can choose $h_j = h_k$ and it works, since $h_k(\bar{x}_i)$ is consistent with some tuple in S_j.
  - Thus it is enough to choose h_j only for nodes in T. We claim that it is enough to choose h_j so that it is consistent with the homomorphism associated to the parent of X_j. The proof is basically the same as in the proof of proposition 19.2 in the book.

  Hence we have shown for all i that $S_i \subset \pi_{\bar{x}_i}(S_1 \bowtie \ldots \bowtie S_n)$, and the inverse inclusion being always true, this concludes.

  Other solution, prove by induction on the join tree structure that (local) implies (global)

Exercise 2. Start with $S_i = q_i(D)$, and iterate the operation of iterating over all i and perfoming a semi-join of S_i on all other S_j, successively.

This works because if $A \subset B$ and $C \subset D$ and $A = A \ltimes C$, then $A \subset B \ltimes D$.

Exercise 3. $q = R(a,b)R(b,c)R(c,a)$

R[1]	R[2]
1	2
2	1

Yannakaki's Algorithm

Exercise 4. Sort both and do a joined scan over $q(D)$ and $q'(D)$, which computes in $\mathcal{O}(x\log(x) + y\log(y))$. Then to compute

$$q_S = \texttt{Answer}(\bar{y}) : -R_{j_1}(\bar{u}_{j_1})\ldots$$

we iteratively compute

$$\begin{aligned}&R_{j_1}(\bar{u}_{j_1})(D)\\ &R_{j_1}(\bar{u}_{j_1})(D) \ltimes R_{j_2}(\bar{u}_{j_2})(D)\\ &\vdots\end{aligned}$$

Exercise 5.

- ⋆ First we show that the first pass can be computed in the desired complexity. We can first compute all $q_s(D)$. A single computation takes $\mathcal{O}(||q_s|| \cdot ||D|| \cdot \log(||D||))$ so the sum of the computations for all s is $\mathcal{O}(||q|| \cdot ||D|| \cdot \log(||D||))$. Remains to compute the

$$Q_s(D) = \bigcap_{1 \leq i \leq p} q_s(D) \ltimes Q_{s_i}(D)$$

We recall that $||q_s(D)|| \leq ||D||$ and $||Q_{s_i}(D)|| \leq ||q_{s_i}(D)|| \leq ||D||$, implying that $Q_s(D)$ can be computed in $\mathcal{O}(p \cdot ||D|| \log ||D||)$. The sum of thoses computations for all s asks to control the size of each p, which sum to the number of nodes in the join tree – that is $\mathcal{O}(||q||)$. In the end, all Q_s can be computed in $\mathcal{O}(||q|| \cdot ||D|| \cdot \log(||D||))$.

- ⋆ Note that if $\bar{x}$ is contained in a single node of the join tree, call it t, which can be detected in $\mathcal{O}(||q||)$, then we can root the join tree at t and execute only the first pass of the algorithm. Then, the computed $Q_t(D)$ is already $q(D)$ (up to a projection onto $\bar{x}$ which can be done in $\mathcal{O}(||Q_t(D)||) = \mathcal{O}(||D||)$). Hence in that case, the total running time of the algorithm is $\mathcal{O}(||q|| \cdot ||D|| \cdot \log(||D||))$.

- ⋆ Next we show that the second pass can be computed in the desired complexity. This is easy : $||\mathrm{A}_s|| \leq ||Q_s(D)|| \leq ||D||$, where we get that A_s can be computed in $\mathcal{O}(||D|| \log(||D||))$. In the end, the whole second pass can be computed in $\mathcal{O}(||q|| \cdot ||D|| \cdot \log(||D||))$.

- ⋆ Next we show that in the second pass of the algorithm, the sets A_s that are computed are the answers on D to the queries

$$A_s(\bar{y}_i) :- R_1(\bar{u}_1) \ldots R_n(\bar{u}_n)$$

that is, we prove theorem 20.3 of the book. Note that for clarity we write A_s for the sets that are computed by the algorithm and A_s for the queries defined above. We show the result by induction on the structure of the tree. Clearly for the root, this is true. Now suppose $\mathrm{A}_s = A_s(D)$ and let s' be a child of s. If $\bar{a} \in A_{s'}(D)$ then clearly $\bar{a} \in \mathrm{A}_{s'}$ (indeed $A_{s'} \subseteq Q_{s'}$). The difficulty lies in showing that $\mathrm{A}_{s'} \subset A_{s'}(D)$.

Let $\bar{a} \in \mathrm{A}_{s'} = Q_{s'}(D) \ltimes A_s(D)$, from which we get a homomorphism $h_{s'}$ witnessing $(A_{Q_{s'}}, \bar{y}_{s'}) \to (D, \bar{a})$. By definition, there is a tuple $\bar{b} \in A_s(D)$ that is consistent with $\bar{a}$. This in turns means there is a homomorphism h witnessing $(A_q, \bar{y}_s) \to (D, \bar{b})$. Finally, define h' by completing $h_{s'}$ with h. We claim that h' witnesses $(A_q, \bar{y}_{s'}) \to (D, \bar{a})$. Clearly we only need to show that it witnesses $A_q \to D$. Consider some atom $R_i(\bar{y})$ in q. We write $\bar{y} = \bar{y}_1 \cup \bar{y}_2 \cup \bar{y}_3$ where

  - — $\bar{y}_1$ are the variables of $\bar{y}$ that are not in $Q_{s'}$
  - — $\bar{y}_2$ are the variables of $\bar{y}$ that are shared between s' and s
  - — $\bar{y}_3$ are the variables of $\bar{y}$ that are in $Q_{s'}$ but not in s'. Note that then those variables do not appear anywhere else but in the subtree rooted in s', because of the join tree property.

Because of the join tree property, $\bar{y}_1$ and $\bar{y}_3$ cannot be simultaneously non empty. Moreover, h and $h_{s'}$ are equal over $\bar{y}_2$. This shows that h' is equal to either h or $h_{s'}$ over $\bar{y}$, whence we get that $R_i(h'(\bar{y})) \in D$.

To summarize, we have shown that $\bar{a} \in A_{s'}(D)$, which proves that $\mathrm{A}_{s'} = A_{s'}(D)$.

- ⋆ Finally, we must bound the complexity of the third pass. There is some ambiguity as to what it means to project X onto a dimension that is not present in X, i.e. we compute $\pi_{s\cup\bar{x}}(O_s(D) \bowtie O_{s_j}(D))$ but we are not certain that all variables of $\bar{x}$ appear in $s \cup s_j$. In that case, I understood that the variables of $\bar{x}$ that are not represented are ignored. If s is a node of the join tree, let $\bar{Y}_s$ be the variables' subset of $\bar{x} \cup s$ that appear in the subtree rooted in s. Define the queries
$$O_s(\bar{Y}_s) :- R_1(\bar{u}_1) \dots R_n(\bar{u}_n)$$
We first note that the O_s that the algorithm computes are the $O_s(D)$. Indeed, thanks to the join tree property, $\mathrm{O}_s = \pi_{s\cup\bar{x}}(A_s(D) \underset{s_j \text{ child of } s}{\bowtie} O_{s_j}(D))$, i.e. we can defer the projection to the very end. Thus $O_s = A_s(D) \bowtie \pi_{s\cup\bar{x}}(\underset{s_j \text{ child of } s}{\bowtie} O_{s_j}(D))$. In particular $O_s \subseteq A_s(D) \bowtie \pi_{\bar{x}}(\underset{s_j \text{ child of } s}{\bowtie} O_{s_j}(D))$ whence we get that $||O_s(D)|| = \mathcal{O}(||D|| \cdot ||q(D)||)$.
We also notice that in the course of its computation, O_s only ever increases, and thus its size is always bounded by $||O_s(D)||$.
And here I am stuck...

Acyclicity of the Core

Exercise 6. $q(y_1, \dots, y_n) = \exists x \exists z R(x, y_1) \dots R(x, y_n) R(z, y_1) \dots R(z, y_n)$
Or a 2 cycle and any $2n$ cycle

Exercise 7. If R_i in q is mapped to R'_j in q', then define S_i as such : for every $\bar{a} \in S'_j$, deduce what $\bar{x}_i$ is mapped to through h, call it $\eta_i(\bar{a}) = (\eta_{x_{i_1}}(\bar{a}), \dots, \eta_{x_{i_p}}(\bar{a}))$ and add that to S_i, where h witnesses $q \to q'$. If R_i is only mapped to constants, add that one tuple of constants to S_i.

Then take $\bar{u} \in S_i$, choose some $j \neq i$, its associated $\bar{a} \in S'_{i'}$ and $\bar{b} \in S'_{j'}$ consistent with $\bar{a}$. Finally, note $\bar{v} \in S_j$ associated with $\bar{b}$. We want to show that $\bar{u}$ and $\bar{v}$ are consistent.

If $x_{i_k} = x_{j_l}$ is mapped to a variable, it works because $\bar{a}$ and $\bar{b}$ are consistent, if it is mapped to a constant then it works directly. In short, it works.

Conjunctive Queries with Equational Atoms