technische universität dortmund

Can We Probabilistically Generate Uniformly Distributed Relation Instances Efficiently?

Joachim Biskup Marcel Preuß

Technische Universität Dortmund
Germany

DBSEC 2020, August 26, Lyon-online

Problem: Can We ... ?

Simplified Single-FD Scenario and Probabilistic Instance Generation Task

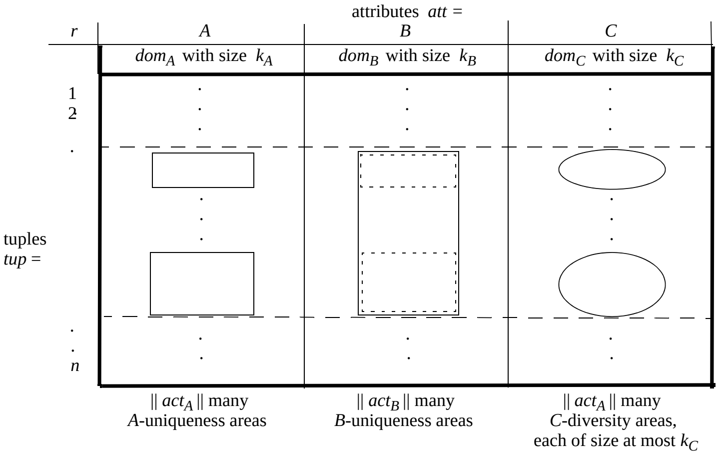
Inputs

$(R(\{A, B, C\}), A \rightarrow B)$	relational schema with **one functional dependency**
$k_{att} = \|dom_{att}\| \geq 2$	size of domain for attribute $att \in \{A, B, C\}$
$n > 0$	required size (number of tuples) of an instance

Output

r	**relation instance as an (unordered, duplicate-free) set, to be generated with uniform probability distribution**

Uniqueness and Diversity Areas for Single-FD $A \to B$



Combinatorial Analysis

Number of Single-FD Instances: n-ak_A-k_C-Representations

$$\sum_{\lceil \frac{n}{k_C} \rceil \leq ak_A \leq \min(k_A, n)} \quad \sum_{\substack{1 \leq k \leq n\,, \\ (s_1, m_1), \ldots, (s_i, m_i), \ldots, (s_k, m_k): \\ 1 \leq s_i < s_{i+1} \leq k_C\,; \\ 1 \leq m_i \leq n\,; \\ n = \sum_{i=1}^{k} s_i \cdot m_i\,; \\ ak_A = \sum_{i=1}^{k} m_i}} \left[\prod_{i=1}^{k} \binom{k_A - \sum_{1 \leq j < i} m_j}{m_j} \cdot k_B^{ak_A} \cdot \prod_{i=1}^{k} m_i \cdot \binom{k_C}{s_i} \right]$$

- $1 \leq k \leq n$ — number of different sizes of A-uniqueness areas
- $(s_1, m_1), \ldots, (s_i, m_i), \ldots, (s_k, m_k)$ — sequence of such sizes with their multiplicity
- $1 \leq s_i < s_{i+1} \leq k_C$ — such sizes strictly ordered, bounded by cardinality of dom_C
- $1 \leq m_i \leq n$ — such multiplicities bounded by overall size of instance
- $n = \sum_{i=1}^{k} s_i \cdot m_i$ — such multiplied sizes sum up to overall size of instance
- $ak_A = \sum_{i=1}^{k} m_i$ — such multiplicities sum up to size of selected active A-domain

technische universität dortmund

n-ak_A-k_C-Representations

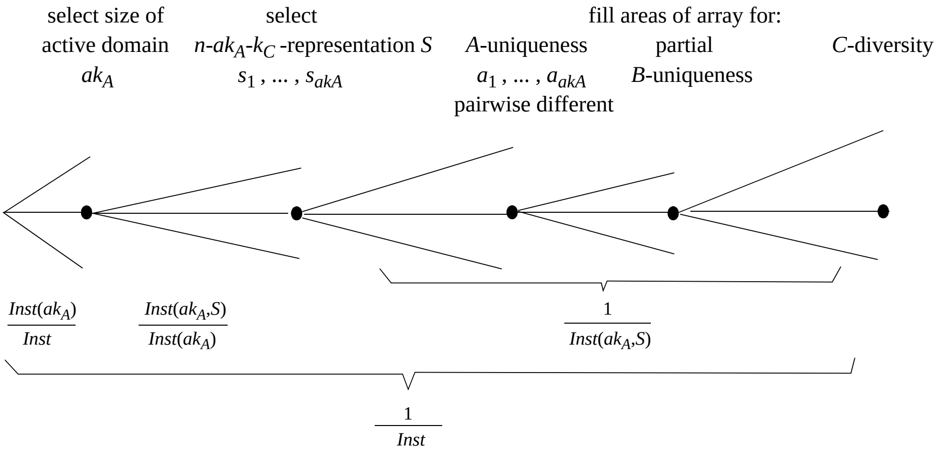
tuples/lines	uniqueness areas	sizes	multiplicities	multiplied sizes
1				
s_1	$\boxed{}$	s_1	m_1	$s_1 \cdot m_1$
$s_1 \cdot m_1$	$\boxed{}$	s_1		
	$\vdots$			
	$\boxed{}$	s_k	m_k	$s_k \cdot m_k$
n	$\boxed{}$	s_k		
			$\Sigma = ak_A$	$\Sigma = n$

Sophisticated Probabilistic Generation Procedure

Probabilistic Generation Procedure: Step I. Preprocessing

1. determine and list all possible sizes ak_A of an active domain act_A for attribute A
2. for each listed ak_A: (solve Restricted Integer Partition Problem [Euler 1741], i.e.,) determine and list all possible n-ak_A-k_C-representations S
3. for each listed ak_A, for each listed n-ak_A-k_C-representation S: calculate and keep the number $Inst(ak_A, S)$ of complying instances
4. for each listed ak_A: calculate and keep the number $Inst(ak_A)$ of complying instances
5. determine and keep the number $Inst$ of all instances
6. annotate each listed size ak_A and each listed n-ak_A-k_C-representation S with probabilities $Inst(ak_A)/Inst$ and $Inst(ak_A, S)/Inst(ak_A)$, respectively

Probabilistic Generation Procedure: Step II. Generation with Probabilities



technische universität dortmund

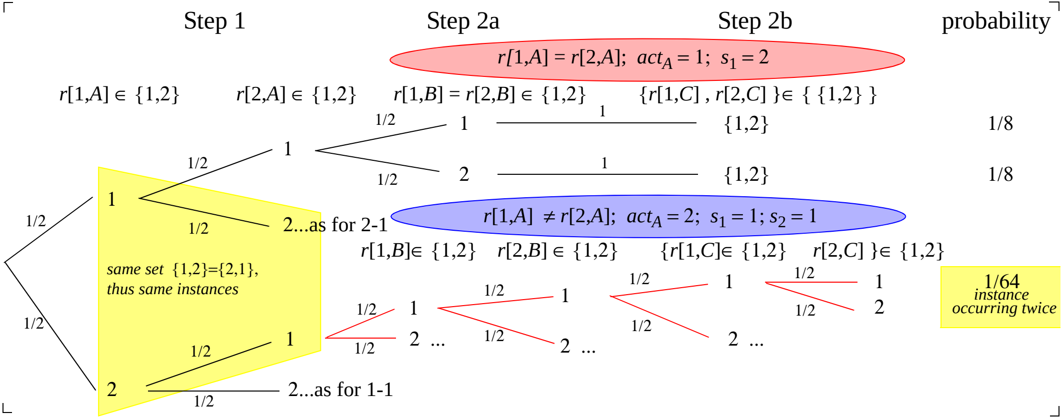
Probabilistic Generation Procedure: Outline of Time Complexity

- **Part I** (performed only once)
  - essentially solving the *Restricted Integer Partition Problem*: exponential

- **Part II** (optimizable if few expected collisions)
  - applying tools based on a standard *pseudo-random generator*
    - *give-next-element* operation: constant
    - *shuffle* operation: linear
    - adaptation of pseudo-randomness: linear
  - *selection of different values* by the following alternatives:
    - repeat the give-next-element operation on a collision
    - shuffle an array representation of the pertinent set of options and extract
  - in total: quadratic (plus the time for adaptations)

Failure of Naive Generation Procedures

"First Choose A-Values, then B-/C-Values" with $n = 2$ and $k_{att} = 2$



Conclusions and Open Answer

technische universität dortmund

Summary of Current Achievements

- systematic counting method for relation instances of required size for one functional dependency and given domain sizes
- identification of the failure of naive approaches to probabilistic instance generation: “local uniform selections” do not necessarily lead to “global uniformity”
- design and verification of a sophisticated probabilistic generation procedure with uniformly distributed outputs:
  1. combinatorially adapted probabilities, to select a template structure
  2. uniform probabilities, to select actual values
- open answer to our question, due to challenging interactions of requirements:
  - by the database schema
  - for the probability distribution of the outputs
- even more challenging for more general scenarios