UNIVERSITY OF SOUTHAMPTON

Faculty of Physical Sciences and Engineering

School of Electronics and Computer Science

Analysis of the complexity of the query containment problem under set and bag semantics

By

AYTAC GOKCE (ag1n16)

Completed on 7th of September 2017

A dissertation submitted in partial fulfilment of the degree of

MSc. Data Science

By examination and dissertation

Supervisor: Dr Pawel Sobocinski

Second Examiner: Dr Gennaro Parlato

ABSTRACT

Conjunctive queries carry a lot of weight with most of the sub branches of computer science, especially, database theory. The conjunctive query containment is ultimate algorithmic issue in database theory. The query containment problem for conjunctive queries under set semantics has been widely solved by some prominent researchers. However, the problem is still open to date for conjunctive queries under bag semantics, namely multisets (duplications for tuples are not eliminated unlike set semantics) which is somewhat significant for SQL. The main purpose of this paper is to survey the complexity of the problem under set semantics and bag semantics by observing theorems improved until now.

ACKNOWLEDGEMENTS

I want to thank to Dr. Pawel Sobocinski, who is my supervisor, for his advices, supports, and most importantly suggesting my thesis topic.

I want to thank to Dr. Gennaro Parlato for his recommendations.

Table of Contents

INTRODUCTION

There have been a lot of investigations in database theory for a great deal of time. In this period, many achievements have been witnessed. Despite the successes, some considerable problems have been occurring and withstanding against resolutions. Query containment problem is considered as one of the most popular amongst them. The problem firstly came up by Chandra and Merlin approximately four decades ago. The problem inquires for given queries QA and QB, is $QA \subseteq QB$ accurate? To put it in other words, is $QA(DB)$ included in $QB(DB)$ for each database DB? [1]. The query containment has related to numerous different scopes comprising query optimization, constraints checking, query processing so on and so forth. However, in this project I analysed the query containment problem under set and bag semantics based on the what people have done so far.

During several years, many types of research have comprehensively examined the query containment problem for distinct operators of conjunctive queries under set semantics. Initially, Chandra and Merlin put forward that the query containment problem for conjunctive queries is NP-Complete by reducing 3-Colouribility. Afterwards, Sagiv and Yannakakis examined the problem for unions of conjunctive queries with the aid of Chandra and Merlin's theorem, and they found out that query containment problem for unions of conjunctive queries is NP-Complete. Following that, Klug in 1988 and van der Meyden in 1992 verified $\prod_2^p$ −completeness for the query containment for conjunctive queries with arithmetic operations.

Looking at the bag semantics, Chaudhuri and Vardi put forward the query containment problem under bag semantics in 1993. According to their study, the problem under bag semantic is $\prod_2^p$ −hard. Subsequently, Ioannidis and Ramakrishnan investigated the problem for unions of conjunctive queries under bag semantics, and they advocated that the problem is undecidable. For conjunctive queries with inequalities under bag semantics, Jayram, Kolaitis and Vee brought forward the problem is undecidable. In addition, some following improvements specifying some exclusive sub classes of conjunctive queries for query containment were witnessed. In 2010, Afrati, Damigos and Gergatsoulis worked on conjunctive queries without projection and self-join. Kopparty and Rossman investigated a large class of Boolean queries on graphs.

Although the complexity of the query containment problem has been sorted out on a large scale under set semantics, we cannot say the same for bag semantics. Despite the considerable efforts, it still maintains open to date. The essential part of this paper is to survey the complexity of the containment problem based on the findings so far.

CONJUNCTIVE QUERIES

Conjunctive queries are the plainest query languages that could be defined on a database [1] [2]. However, they are involved in most of query languages with their somewhat expressive power. Especially, conjunctive queries stand for so elementary attributes of relational algebra which are SELECT, PROJECT, and JOIN.

There exist several ways to express a conjunctive query. Using Datalog notation a conjunctive query can be written as; $Q(a_1, a_2, \dots, a_k) : -R_1, R_2, \dots, R_N$ [2].
A query $Q(\mathrm{x}) : -\ Worksfor("Accounting", y), Worksfor(x, y)$ is represented in SQL as

```
SELECT DISTINCT d2.name

FROM Worksfor d1, Worksfor d2

WHERE d1.name=" Accounting" AND
d1.department= d2.department
```

To write as the form of first order logic which is provided by a set of formulae might be consisted of atomic formulae making use of existential quantification ∃ and conjunction ∧;

$Q\ (a_1, a_2, \dots, a_k) : \exists_{(a_{k+1}, \dots, a_m)}. R_1, R_2, \dots, R_N$ in which $(a_1, a_2, \dots, a_k)$ are free variables, $(a_{k+1}, \dots, a_m)$ are bound variables, and $R_1, \dots, R_N$ are atomic variables [1]. As defined above, conjunctive queries could state major part of queries taken part in SQL. For example,

(Employee, Department). ∃ (Employee2, Name). Works (Employee, Department) ∧ Gender (Employee, 'female') ∧ Works (Employee2, Department) ∧ Gender (Employee2, 'male') ∧ is (Employee, Name)

Conjunctive queries obtained much more significant abilities with the aid of their extensions. The extensions are respectively unions of conjunctive queries which are identical with positive relational algebra, conjunctive queries with inequalities and conjunctive queries with arithmetic operations.

Unions of Conjunctive Queries (UCQ)

A UCQ is a query that can be written as $(Q_1 \cup\ Q_2 \cup \dots\ Q_N)$ in which every Q is a conjunctive query. As an instance, $Q = \ Q_1 \cup\ Q_2,\ Q_1(a, b) = R(a, c), (c, b)$, and $Q_2(a, b) = R(a, c), R(c, d), R(d, b)$ is a UCQ those paths of length are 2 or 3 [3] [4] [5]. We can give the below example to show union in the SQL.

```
SELECT Name, Surname FROM Students
WHERE Surname='Clarke'
UNION ALL
SELECT Name, Surname FROM Teachers
WHERE Surname='Clarke'

ORDER BY Name;
```

Conjunctive Queries with Inequality

If inequality is inserted to conjunctive queries, $CQ^{\neq}$ (conjunctive query with inequality) is attained. To give an example; $Q\ (a, b)$: $\exists z\ \exists w.\ (\exists\ (a, c) \wedge \exists (c, b) \wedge \exists (a, d) \wedge \exists (d, b) \wedge c \neq d)$ the query returns distinct paths of length 2 [6] [7] [5]. We can give the below example to show inequality in the SQL.

```
SELECT *
FROM Students
WHERE Name != 'Michael'
```

Conjunctive Queries with Arithmetic Operations

On condition that operators are inserted to conjunctive queries, $CQ<$ (conjunctive query with arithmetic operations) is acquired. For instance, $Q\ (a, b)$: $\exists z\ \exists w.\ (\exists\ (a, c) \wedge \exists (c, b) \wedge \exists (a, d) \wedge \exists (d, b) \wedge a < b, b < c, c < d)$ the query returns paths of length 4 [8] [5]. We can give the below example to show an arithmetic operation in the SQL.

```
SELECT COUNT (StudentId), Department
FROM Students
GROUP BY Department
HAVING COUNT(StudentId) >= 75;
```

SET vs BAG (MULTISET) SEMANTICS

On the basis of the set theory, relations in a database and the outputs of implemented queries are sets of tuples. The main feature of the set semantics is that they exterminate duplications [9].

Under bag semantics, duplications are not eliminated. That is because duplications are prominent for aggregations in real relational databases [10]. It could be thought as every tuple has an integer which represents its multiplicity. According to this concept, each relation has a set of tuples of the form $p(A_1, A_2, \dots, A_N; [m])$, where p is the predicate, $A_1, A_2, \dots, A_N$ are constant of domain and m is multiplicity (it can be called as the number of copies of tuples either) [9].

In the relational algebra, the operators are enlarged to work on bags by multiplicities. RDBM (Real Database Management Systems) is based upon bags instead of sets in order to carry out bag

semantics. In bag semantics, we might specify every relational operator differently from set semantics (SELECT, PROJECT, and JOIN take part in both bag and semantics) [9].

Bag semantics independently operates every tuple, that is, it is possible to see a tuple more than once in the final relation. To make it clear, following examples can be shown for projection and selection.

R			$\pi_{X,Y}(R)$		$\sigma_{Z='d'}(R)$		
X	Y	Z	X	Y	X	Y	Z
a	b	c	a	b	a	b	d
a	b	d	a	b	b	c	d
b	c	d	b	c	b	c	d
b	c	d	b	c			

When we look at the union, intersection, and difference of bag semantics, if a tuple takes part w times in relation R and k times in relation S, then the tuple takes part $w + k$ times in the union $R \cup S$, min $\{w, k\}$ times in the intersection $R \cap S$, max $\{0, w - k\}$ in the difference $R - S$ [9].

R		S		$R \cup S$		$R \cap S$		$R - S$	
X	Y	X	Y	X	Y	X	Y	X	Y
a	b	a	b	a	b	a	b	b	c
a	b	a	b	a	b	a	b		
b	c	a	b	a	b	b	c		
b	c	b	c	a	b				
		b	d	a	b				
				b	c				
				b	c				
				b	c				
				b	d				

When we look at the perspective of SQL for bags, SQL is based upon bag semantics rather than set semantics as defined above. For this reason, getting out of duplications usually calls for an expensive sort. Notwithstanding, SQL ensures an effective method to emulate set semantics using DISTINCT keyword, which transforms a bag relation to the matching set one. To exemplify,

```
SELECT DISTINCT R.X, S.Y
FROM R, S
WHERE R. Z=S.Z
```

Chaudhuri and Vardi denoted that the main rationality for using bag semantics for SQL can be explained for two reasons. First, eliminating of duplications may take a lot of time. The other, aggregate queries (including COUNT, SUM, MIN etc.) are sensible to the multiplicity.

COMPUTATIONAL COMPLEXITY THEORY

Computational Complexity Theory is one of the essential investigation area of the theoretical computer science, its basic objective is to categorize computational problems with respect to their innate difficulty and to establish connections between computational tasks [11]. Last three decades have been witnessed considerable enhancement in the field of the theory. Initially it investigated the computation of well-defined functions on an entirely represented input by a single and deterministic machine, which still goes on to be a research area. By the agency of some computational models like interactive, randomized, and distributed computation, the scope then transformed into exploration a somewhat wide range of computational tasks [12].

Should its resolution necessitate notable sources, a problem is admitted as intrinsically arduous irrespective of algorithms made use of. At that point, complexity theory takes advantages of mathematical models of computation in order to examine these complicated problems as well as define the quantity of sources required to sort them out. Likewise, further improvements in Mathematics might be highly affected by computer science, namely, the connection between these fields is mutual. In addition to the Mathematics (especially Discrete Mathematics, Algebra and Analysis), and computational complexity theory are connected to some other fields on a large scale such as Physics and Economics [13] [14].

Handling the advancement of computational complexity theory, the primary work in complexity theory specified the fundamental computational models from the mid of 1960s to 1970s. Computational complexity was founded as a stand-alone mathematics field by the Turing Machine complexity theory and the axiomatic complexity theory. Before the early 1970s, the recognition of polynomial time computability with feasible computability and the finding of the NP-complete problems gave rise to $P \stackrel{?}{=} NP$ question as the principal problem of the complexity theory. Until to the mid of 1980s, studies in computational complexity widened at a considerable extent. Diverse other models were put forward alternatively to the conventional deterministic ones whereby acquiring more proper classifications for the algorithmic problems. In the mid of 1980s, it is possible to say that very profound and interesting developments were witnessed so that complexity theory has been particularly associated with the investigation of the feasible complexity classes below PSPACE. Accordingly, the new notions of source bounded reductions as well as complete languages for inherent complexity classes were alterations of main recursive functions theory, however, it started a tremendously extensive and interesting area of study [14] [12] [13].

COMPLEXITY CLASSES

Complexity classes are series of computational problems of relevant source-based complexity. Ultimately, these classes might be pointed out by determination of three parameters [15] [14].

- A sort of computational problems: Undeniably, vast majority of problems bound up with decision problems. Nevertheless, classes of other problems such as search problems, promise problems are also highly notable part of complexity classes.
- A pattern of computation: It is related to belonging to either uniform or non-uniform models. Generally accepted, the most popular computational model is deterministic Turing Machine, however, several complexity classes are founded on Boolean circuits, nondeterministic Turing Machine.
- A complexity measure or functions (a series of function or a function): Complexity classes are categorized by considering time complexity, space complexity and polynomial time computations.

Time and Space Definitions

As for each function : $N \to N$:

$x \in TIME\left(f(n)\right)$ in case a Turing Machine *TM* is used for the computation, and presumably it has input tape and output tape for $p \geq 1$. The machine *TM(s)* takes as many steps as $f(|s|)$ where for each input *s,* $|s|$ defines the length of *s* in bits [16] [12].

In a similar way, $x \in SPACE\left(f(n)\right)$ in case a Turing Machine *TM* is used for the computation, and presumably it has input tape and output tape for $p \geq 1$. The machine *TM(s)* uses at most $f(|s|)$ squares of its work tapes for each input *s.* A square refers to being scanned pending computation [16] [12].

Time and space complexity can be specified for functions either.

A function $f \in FTIME\left(t(n)\right)$ in case a Turing Machine *TM* is used for the computation and presumably it has input tape and output tape for $p \geq 1$. *TM(s)* records *f(s)* on its output tape. Besides, the machine takes as many steps as $t(n)$ [16] [12].

Likewise, $f \in SPACE\left(t(n)\right)$ in case a Turing Machine *TM* is used for the computation and presumably it has input tape and output tape for $p \geq 1$. *TM(s)* records *f(s)* on its output tape. Besides, *TM* does not use more than *t(s)* squares on its work tapes [16] [12].

Polynomial Time and *P* Class

A Turing Machine is named polynomial, on condition that a function $f(n)$ exists and the time $t(n) = O(f(n))$. To put it other words, assuming that *c* is a constant so that the time demand of *TM* is indicated as $O(n^c)$.

With respect to class of decision problems, subsequent class is widely regarded as *efficiently solvable*. A function that belongs this class is computable in polynomial time. It is also worth mentioning that a decision problem allows only yes or no answers [16].

$f \in P, where \equiv \cup_{k\geq 0} TIME(n^k)$ [16]

The class P takes part in the class of search problems which could be resolved efficiently as defined above. A problem is taken into account as feasible providing that a solution whose time complexity is in a polynomial rate, however, if the solution's rate of growth is super-polynomial on its running process then the solution is considered as unfeasible. This characterization is unreasonable or arguable for some researchers in this concept that whether $O(n^{100})$ time is actually feasible computation in real world, whereas when a problem is to be demonstrated as in P, generally founded time algorithms are much less than n^{100} time algorithm [16] [15].

P is the lowest class comprising *TIME(n),* and it allows modular composition. That means, components of *P* are able to invoke each other as subroutines by virtue of polynomial closure under multiplication and composition so that the resulting programs still exist in *P.* That is a somewhat natural class from the standpoints of programmers, and for efficient programming [12].

THE CLASS NP

The complexity class *NP* purposes to get hold of the set of problems that could be efficiently substantiated. More clearly, the *NP* (stands for nondeterministic polynomial time) is the set of decision problems resolvable in polynomial time by a nondeterministic Turing machine. A language or decision problem is in *NP* class if *x* is given as an input, it might simply be verified that *x* is a 'yes' answer of the problem. The *P* versus *NP* problem which may stand for the most well-known complexity problem inquires whether these two classes are equivalent or not. The complexity class *NP* is also associated with to the complexity class co-NP which has effectively provable evidences as for the answer 'no'. Another problem whether *NP* is equivalent to *co-NP* is also ostensible in complexity theory [11] [17].

Several definitions can be given for *NP* but in the simplest term:

$$NP = U_C\ NTIME(n^c)$$

To rewrite the formula in more technical concept of non-determinism: $x \in \{0,1\}^*$, a language $L \subseteq \{0,1\}^*$, *a* is an integer, and a polynomial time relation *M* so that

$x \in L \leftrightarrow \exists u. |u| < |x|^a . M(x,u)$ [12]

At that case, the u is 'guess', which M 'checks', and it is frequently named that u is a certificate for x with regards to L and M [16] [12].

Reducibility

Before handling NP complete problems, it is very useful to mention about reducibility that is a very common concept to explore relationships between problems. Reductions are the core part of the theory of NP completeness.

Polynomial – time Reducibility: X is polynomial time reducible to Y, or in other words $X \leq_p Y$, if a polynomial time function $f: \{0,1\}^* \rightarrow \{0,1\}^*$ for each $\in \{0,1\}^*, x \in X \; if \; f(x) \in Y$.

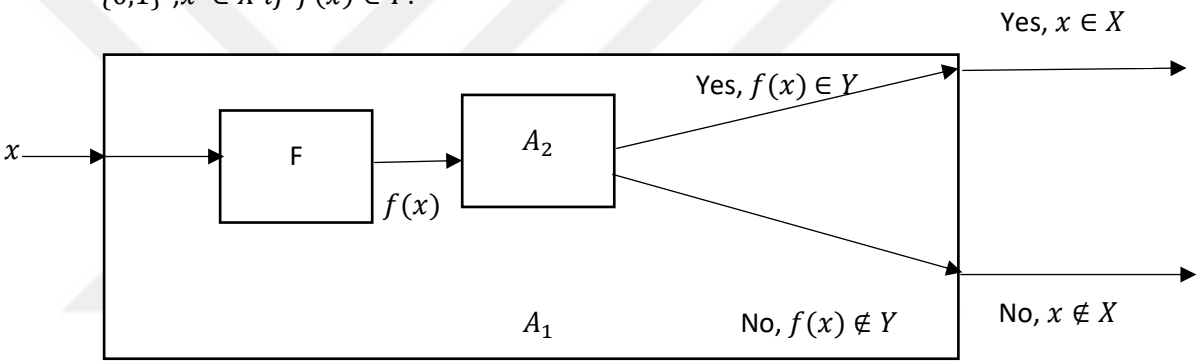


FIGURE 1 POLYNOMIAL TIME REDUCIBILITY [30]

1. A problem *X* is *Cook (Turing)* reducible to another problem *Y* ($X \leq_t Y$) if there is a polynomial time solvable oracle *M* then M^Y decides *X.*

2. A problem *X* is *Karp (Levin, many-one)* reducible to another problem *Y* ($X \leq_m Y$) if there is a poly-time function p then $z \in X \leftrightarrow p(z) \in Y$.

3. A problem *X* is log space reducible to another problem *Y* ($X \leq_l Y$) if there is a log space function l then $z \in X \leftrightarrow l(z) \in Y$.

Loosely speaking, these correlations are transitive, albeit showing transitivity of $\leq_l$ is a bit complicated. Such that, $X \leq_l Y \Rightarrow X \leq_m Y \Rightarrow X \leq_t Y$
According to given specification, if $X \leq_t Y \Rightarrow Y \in \boldsymbol{P}$, in that case, $X \in \boldsymbol{P}$ [12] [17].

NP COMPLETENESS

On the basis of the complication of P versus NP question, it could not be assumed to verify that a hard problem Q in NP is not in P (unquestioningly). The most optimum thing can be supposed is a conditional argument that Q is not in P, according to the presumption that NP is different from P. On the other hand, if Q is in P, then any other problem Q' exist in NP based on the $NP = P$. The principle of the theory of NP-completeness is that to show any problem in NP is polynomial time reducible to Q as potential way of verifying such an argument [13] [11] [18].

A problem Q is ***NP-hard*** providing that a polynomial time algorithm for Q could show a polynomial time algorithm for each problem in NP. Assuming that if a particular NP-hard problem can be rapidly resolved, at that case, any problem whose solution is simple to understand might be rapidly solved by using the solution to that one special problem as a subroutine. A problem is ***NP—complete*** if it is NP-hard and a component of NP. More technically, a set X is *NP-hard* with regards to $\leq_l$ if $\forall\, Y \in NP.\, (Y \leq_l X)$. X is *NP-complete* if $X \in NP$ as well as X is *NP-hard.* Commonly accepted, NP-complete problem are the hardest problems in NP. Np-completeness essentially refers to expressiveness. The definition $X \leq_l Y$ indicates that there exists a way to state computations regarding X as computations regarding Y so that; the expression X is *NP-hard* indicates that any effective computation could be stated as a sample of X. Conversely, the definition $X \leq_l Y$ does not have to indicate anything about running times for X and Y, as the expressions of samples X might be merely an exclusive subset of the samples Y. Therefore, expressiveness provides indirect but potent route to specify hard problems in NP [12] [16].

NP-COMPLETE PROBLEMS

Some of the most natural instances of NP-complete problems occur with propositional logic. A Boolean formula over the variables $a_1, a_2 \ldots a_n$ comprise the variables and the logical operators. As an example, $(x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$ is a Boolean formula which is true iff most of variables x, y, z are true. A Boolean formula over variables is in CNF (Conjunctive Normal Form) providing that it consists of AND of OR's of variables or their negatives. As an example; $(\bar{a}_3 \vee a_1) \wedge (\bar{a}_2 \vee a_3) \wedge (\bar{a}_1 \vee a_2)$. Let ϕ be a Boolean formula over variables $a_1, a_2 \ldots a_n$, and let ω be $\in \{0,1\}^n$, and then $\phi(\omega)$ remarks the value of ϕ as the variables of ϕ are designated the value ω. The formula ϕ is *satisfiable* (loosely named as SAT) if there is some assignment ω so that $\phi(\omega)$ is true. In the opposite case, ω can be said *unsatisfiable* [15].

The Cook-Lewin Theorem (1971): *SAT is NP-Complete.*

By looking at the description of NP-complete, *SAT* belongs to NP class. That is because a satisfying assignment could serve as a certificate that a formula is satisfiable.

Proof Purpose: It could be showed that any language L in NP is effectively reducible to *SAT*. Let *M* be non-deterministic machine for given language. The function of reduction takes a string w and generates a Boolean formula $\phi_{M,w}$ that simulates *M* on the input w. An assignment to $\phi_{M,w}$ might show a computational way of the NTM machine *M,* so that $\phi_{M,w}$ becomes satisfiable if and only if the machine *M* approves the input w.

Proof: Before proving the theorem, it is worth mentioning the tableau.

#	q_0	w_1	w_2	…	w_n	⊔	…	⊔	#
#									#
#									#
#									#

FIGURE 2: $n^k * n^k$ CELL TABLEAU FOR M ON INPUT W [19]

Let w be input, L be language, and M be non-deterministic machine. Suppose that M decides whether $w \in L$ in n^k steps. The above tableau represents $n^k * n^k$cells for the machine M on input w. A variable can be exemplified as $x_{i,j,s}$ which is true if cell $[i, j]$ is s; else, false. Each satisfying assignment to the variables defines one symbol amongst a state, a tape alphabet, and a # in each cell (φ_{cell}). Accordingly, a Boolean formula should be generated that its parts provide these symbols to comply with an accepting tableau [20].

$$\varphi = \varphi_{cell} \wedge \varphi_{init} \wedge \varphi_{move} \wedge \varphi_{accept}$$

The initial fragment of φ_{cell} inside the brackets lays down 'each cell has at least one element' as a condition whilst latter fragment lays down 'every cell includes no more than one different symbol' [19].

$$\varphi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[\underbrace{\left(\bigvee_{s \in C} x_{i,j,s} \right)}_{each\ cell\ has \ge 1} \wedge \underbrace{\left(\bigwedge_{\substack{s, s' \in C \\ s \neq s'}} (\neg x_{i,j,s} \vee \neg x_{i,j,s'}) \right)}_{each\ cell\ has \le 1} \right]$$

In the formula, $C = Q \cup \Gamma \cup \{\#\}$ in which Γ and Q are the tape alphabet and state set of the machine M in turn.

The φ_{init} part provides that each cell of the first row has a symbol correlating with the initiation configuration of machine M on the input w.

$$\varphi_{init} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}$$

The φ_{accept} part stipulates an accepting configuration to take place in the tableau such that at least one cell is the accept state which is q_{accept}.

$$\varphi_{accept} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}$$

Lastly, the formula φ_{move} controls whether each $2 * 3$ window is *legal* or not based on the transition principle of the Turing Machine. To mention about legality of the window, this condition must be ensured; the actions defined by the machine M's transition function δ are not distorted by the window. To give an illustration d, e and f are the elements of the Γ, q_1 and q_2 are the states of M.

$\delta(q_1, d) = \{(q_1, e, R)\}$, $\delta(q_1, e) = \{(q_2, f, L), (q_2, d, R)\}$

Some of *legal* window examples:

d	q_1	e
q_2	d	f

d	d	q_1
d	d	e

#	e	d
#	e	d

e	e	e
f	e	e

Some of *illega*l window examples:

d	e	d
d	d	d

d	q_1	e
q_1	d	d

e	q_1	e
q_2	e	q_2

f	q_1	e
e	e	q_1

Suppose that P is the cluster of legal (i,j) $windows$, R is the cluster of the entire(i,j) $windows$. In that case, $P_{i,j} \subseteq R_{i,j}$, $|R_{i,j}| = |C|^6 \implies |P_{i,j}| \le |C|^6$, and $|P| = |\sum P_{i,j}| \le |C|^6 * n^{2k}$ equations are provided. This shows us that the cluster of legal windows is not infinite [19].

When looking at the structure of φ_{move}, it proves that all the $2 * 3$ windows are legal in the tableau:

$$\varphi_{move} = \bigwedge_{1 \le i,j \le n^k} (the\ (i,j)\ window\ is\ legal)$$

The formula can be also written in different way by replacing $(the\ (i,j)\ window\ is\ legal)$ part with bottom one that is for each legal $(i,j)\ window$:

$$\bigvee_{\substack{d_1,d_2 \dots d_6 \\ is\ a\ legal\ window}} (x_{i,j-1,d_1} \wedge x_{i,j,d_2} \wedge x_{i,j+1,d_3} \wedge x_{i+1,j-1,d_4} \wedge x_{i+1,j,d_5} \wedge x_{i+1,j+1,d_6})$$

And finally, the Boolean formula φ was acquired as wished $\varphi = \varphi_{cell} \wedge \varphi_{init} \wedge \varphi_{move} \wedge \varphi_{accept}$. Providing that φ is satisfiable, the context $w \in L$ is met. So as to prove that works in polynomial time, the complexity of the reduction might be examined by analysing of φ's size. As mentioned before the tableau is $n^k * n^k$cells that means it includes n^{2k} cells. Suppose that every cell contains t variables, in which t stands for the number of symbols in C, which is $O(n^{2k})$. While each part of φ is figured out separately, the $\varphi's$ total size is seen as $O(n^{2k})$ such that sizes for φ_{cell} ,φ_{init}, φ_{move}, φ_{accept} are $O(n^{2k})$, $O(n^k)$, $O(n^{2k})$, $O(n^{2k})$ respectively. This boundary is adequate to show main objective since it shows that $\varphi's$ size is polynomial in n [20] [12].

Theorem: *3SAT is NP-complete.*

Suppose that the 3SAT problem that takes Boolean formula B which is in CNF form over n variables $x_1, x_2, \dots x_n$ where each clause has three variables. The objective is to obtain an assignment to the literals that satisfy B. The subsequent formula might be given as instance of 3SAT.

$(\overline{x_1} \vee x_2 \vee x_3) \wedge (\overline{x_2} \vee x_4 \vee x_5) \wedge (x_3 \vee x_3 \vee \overline{x_4})$.

Proof: To verify 3SAT is NP-Hard, SAT problem is reduced in polynomial time. Let ϕ be a Boolean formula in CNF as well. If the following modifications are applied for every clause ϕ_i in original clause ϕ :

A	b	c	$B = c \Leftrightarrow (a \wedge b)$
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	0
0	0	0	1

FIGURE 3: A TRUTH TABLE FOR BOOLEAN FORMULA B=$(\bar{a} \vee b \vee \bar{c}) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (a \vee b \vee \bar{c}) \wedge (\bar{a} \vee \bar{b} \vee c)$

- In case that ϕ_i has one term (e.g. $\phi_i = (a)$), then ϕ_i is replaced with $B_i = (a \vee b \vee c) \wedge (a \vee \bar{b} \vee \bar{c}) \wedge (a \vee \bar{b} \vee c) \wedge (a \vee b \vee \bar{c})$ in which b and c are new variables that are not used anywhere else.
- In case that ϕ_i has two terms (e.g. $\phi_i = (a \vee b)$), then ϕ_i is replaced with $B_i = (a \vee b \vee c) \wedge (a \vee b \vee \bar{c})$ in which c is a new variable that is not used anywhere else.
- In case that ϕ_i has three terms (e.g. $\phi_i = (a \vee b \vee c)$), then $\phi_i = B_i$.
- In case that ϕ_i has more than three terms (e.g. $\phi_i = (a_1 \vee a_2 \vee a_3 \vee \ldots \vee a_n)$), then ϕ_i is replaced with $B_i = (a_1 \vee a_2 \vee b_1) \wedge (\overline{b_1} \vee a_3 \vee b_2) \wedge (\overline{b_2} \vee a_4 \vee b_3) \wedge \ldots \wedge (\overline{b_{n-3}} \vee a_{n-1} \vee a_n)$ in which $b_1, b_2 \ldots b_{n-1}$ are new variables that are not used anywhere else.

According to the above expressions, the value assigned to the later defined variables is entirely unrelated. Regardless of what is assigned to them, ϕ_i is equal to 1 iff the sub formula B_i also equals to 1. Therefore, the preliminary clause ϕ equals to 1 iff B equals to 1. Additionally, every sentence goes up in size pending a constant factor at the utmost and the related computations are basic replacements and hence it has been indicated that how to reduce a sample of the SAT problem to a tantamount sample of the 3SAT problem [21].

SOME OTHER NP-COMPLETE PROBLEMS

***Theorem*:** *Independent Set is NP-Complete.*

Given a graph $G = (V, E)$, a set k is an integer. Independent Set problem is whether there exists a set of k vertices of G such that there is no edge in E between them [15]. Independent Set is obviously in NP since a non-deterministic Turing Machine can prove in polynomial time a set of k vertices, and therefore, it is adequate to demonstrate that Independent Set is NP-Complete which will be implemented by reducing 3SAT.

Proof: Suppose that φ is a 3CNF formula with n variables and m clauses, G is a graph of $7m$ vertices as the following. A set of 7 vertices belonging to G is associated with every clause of φ. The vertices grouped in cluster linked with a clause C correspond to the 7 possible partial assignments to the variables in clause (The reason for calling *partial assignments* is that they merely give values for some variables). Other than the internal edges, an edge is inserted between two vertices of the graph providing that they correspond to *inconsistent partial assignments*. In such a way that if two partial assignments ensure the equal value for every

variable they have, they can be regarded as consistent partial assignments. Additionally, we could insert edges between each two vertices existing in the identical cluster [11] [16].

It is straightforward that converting φ into the graph G could be implemented in polynomial time. It can be said that φ is satisfiable iff G includes a clique of size m. Let φ has a satisfying assignment u, and specify a set S of m vertices as the following. With regards to each clause C of φ inserted in the S, the vertex connected with C that corresponds to the limitations of u to the variables. That is because we solely opt for vertices corresponding to the limitations of the assignment u. S does not consist of two vertices corresponding to inconsistent assignments, and therefore S is an independent size of m. Furthermore, assume that S of size m belongs to G. In that case, S will be used to generate a satisfying assignment u for φ that u can be specified as the following. For each $i \in [n]$, if there exist a vertex in the independent set whose partial assignments ensures a value x to u_i, at that case $u_i = a$; otherwise $u_i = 0$ where u_i might take at most a single value. Moreover, every edge has been grouped in each cluster, S could comprise maximum a single vertex in each cluster, and therefore there is a component of S in each of the m cluster. Consequently, according to the description of u, all φ's clauses are satisfied [11] [16].

Theorem**:** *Vertex Cover (VC) is NP-Complete.*

A cluster of vertices U in a graph $G = (V, E)$ is a *vertex cover* on condition that each edge belonging to E includes at least one vertex in U. Within the vertex cover problem, it must be decided if there exists a vertex cover S with K vertices where K is an integer [22, 11].

Proof: According to given a cluster of vertices, it could be checked out whether it is a vertex cover by considering each edge $\in E$, and looking at the size of the cluster is at the very most K. This can be implemented in polynomial time, and hence vertex cover is clearly in NP. So as to show vertex cover is NP complete, independent set will be reduced as the following. An instance of independent set (G, K) is converted into vertex cover instance as $(G, n - K)$ in which n represents the number of vertices in the graph G. A subcluster of S is an independent set of G if and only if the set $V \setminus S$ is a vertex cover of G [22].

- Suppose that S is independent set of G. Let (v, w) be any edge belongs to E. Then, just one of the vertices (v, w) could take part in S. On account of this, at least one of (v, w) is in $V \setminus S$. Consequently, $V - S$ is a vertex cover [22].
- Suppose that $V \setminus S$ is a vertex cover of G, and a pairwise of (v, w) belong to E. At that case, it is not possible to exist an edge between u and v since this edge could not be covered by $V \setminus S$, and hence S is an independent set of G [22].

Theorem: *Clique is NP-Complete.*

A clique in a graph G is a set U of vertices such that, each pair of vertices is linked by an edge in G. To put it other words, there exists an edge each pair of discrete vertices in U [11] [22].

Proof: Looking at a set of vertices, it can be checked whether in polynomial time or not, that is why clique problem is in NP. To show clique is NP complete, independent set will be reduced as the following. An instance of independent set (G, K) is converted into clique instance as $(\bar{G}, K)$

where $\bar{G}$ is the edge-complement of G with the equivalent vertices but entirely the opposite edges. A subcluster of S is an independent set of G if and only if the set S is a clique in $\bar{G}$ [22].

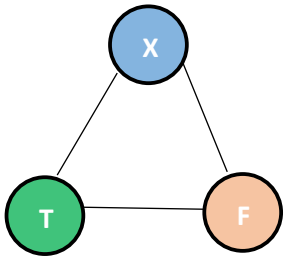
- Suppose that S is independent set of G. There are no two vertices in S that they partake of an edge in G. That means, each pair of vertices in S partake of an edge in $\bar{G}$.
- Suppose that S is a clique in $\bar{G}$ such that, each pair of vertices in S has an edge in $\bar{G}$. Accordingly, there are no two vertices in S that they partake of an edge in G. Therefore, S is an independent set of G.

Theorem: *3-Coloring is NP-Complete.*

The colouring problem on the graphs asks whether there is a way to colour each node with k colours, so that there should be no two connected nodes that have the same colour. Based upon the definition, we can now define the 3-Coloring problem. Given the graph $G = (V, E)$, for each node using at most 3 colours check whether the colour of v is different from the color of w for every edge (v, w) [23] [17].

Proof: 3SAT will be reduced to show 3-Coloring is NP-complete. To reduce 3SAT to 3-Coloring, an appropriate graph G which is 3-colorable if and only if 3CNF formula φ is satisfiable should be generated. Let φ be defined with n variables as $x_1, x_2, \dots, x_n$ and m clauses as $C_1, C_2, \dots, C_n$. The combination of three gadgets will be used to sort out the problem [17] [23] [24].

- The first gadget is used to assign meanings, namely, truth gadget which is a triangle including three vertices as T, S and X that correspond to true, false and other in turn. Seeing that the nodes are neighboring, they all ought to have dissimilar colors (The color given to T will be evaluated as true, given to F will be evaluated as False, given to X will not stand for any special meaning) [17] [23].



- The second gadget consists of a variable and its complement linked to node X in the truth gadget. Therefore, these two nodes must be colored with different colors [17] [23].
- The last gadget can be called as clause *satisfiability gadget* that combines three literal nodes to output node using five non-labeled nodes as well as ten edges as shown below figure. As for a clause $C_i = (a \vee b \vee c)$, the OR of its literals needs to be defined using T, F, X colors. The main objective wanted to be accomplished is to colour output node as T. If literals are coloured F in a 3-coloring, the output node must be coloured F. If one of the literals is coloured T, then the output node is coloured T [17] [23].

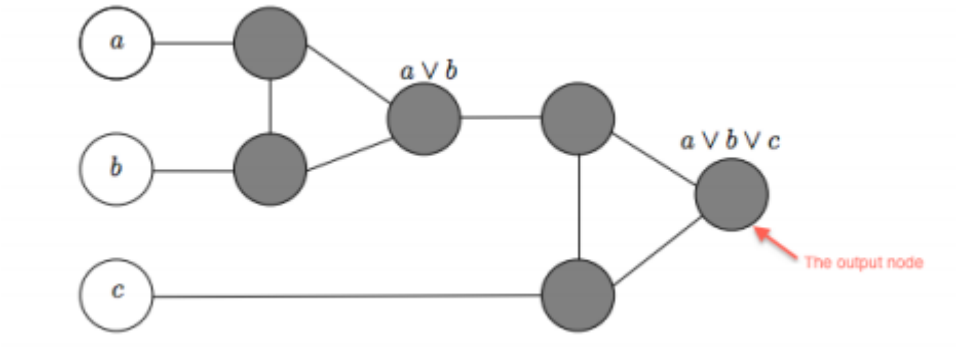


FIGURE 4 CLAUSE SATISFIABILITY GADGET [23]

The last version of the graph G accurately consists of one T node, one F node, one X node, and two nodes for a variable and its complement. To give an illustration, the below figure is a 3-colorable graph that comprises the 3CNF formula$(a \vee b \vee c) \wedge (b \vee \bar{c} \vee \bar{d}) \wedge (\bar{a} \vee c \vee d) \wedge (a \vee \bar{b} \vee \bar{d})$. According to this, one of the 3-coloring complies with the satisfying assignment (where a and c is True, b and d is False) [17].

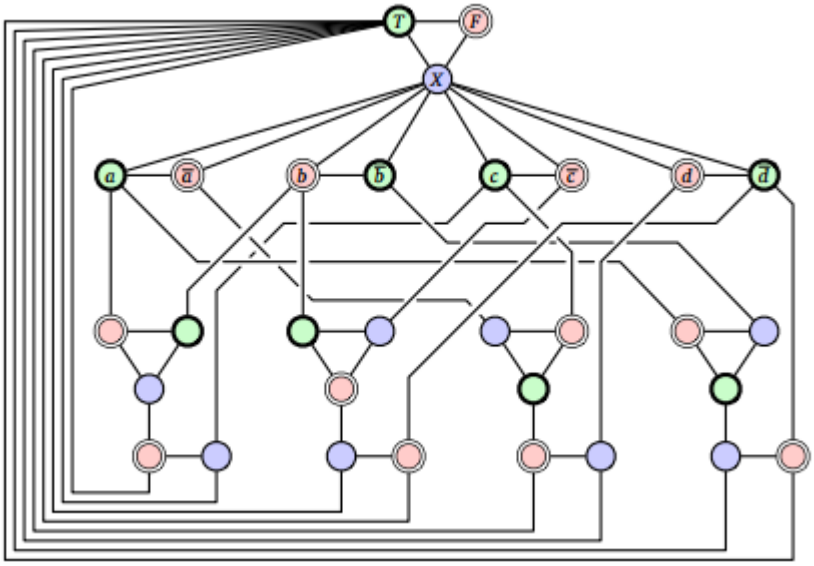


FIGURE 5 FINAL VERSION OF THE 3-COLORING GRAPH [17]

We can now analyze accuracy of the reduction. From the aspect of φ is satisfiable, the graph G is 3-colorable. That is, it is possible to provide a satisfying assignment. Then, at least one of the three literal nodes a, b, c is colored true for each clause C. On the other hand, the graph G is 3-colorable stands for the formula φ is satisfiable. Then, the graph can be coloured by any satisfying assignment. Besides, for any clause $C = (a \vee b \vee c)$, it is not possible to be false for all a, b, c. Finally, it could be easily proved that a graph can be 3-coloured in linear time with the help of crosschecking the endpoints of each edge. Hence, 3-Coloring is in NP-Complete.

THE COMPLEXITY OF THE QUERY CONTAINMENT PROBLEM UNDER SET SEMANTICS

Theorem - Chandra and Merlin (1977): *The query containment problem for conjunctive queries is NP-complete.*

Before the proof, it is useful to touch upon homomorphism and isomorphism theorems that are the essential parts of Chandra and Merlin theorem. A homomorphism from graph G to graph H is a mapping f which is amongst the vertex of sets of two graphs that maps contiguous vertices to contiguous vertices. The function can be indicated as $f: V(G) \longrightarrow V(H)$, and for every edge $vw \in E(G)$, there exists and edge for H like $f(v)f(w)$. A homomorphism from G to complete graph K_n is identical to k-coloring of the graph G. Thus, we can say that graph homomorphism is a generalization of graph coloring. Homomorphism between two graphs is showed as $h: G \longrightarrow H$ [1].

An isomorphism is bijective morphism that maintains relations and sets amongst elements [25], which is also homomorphism. We might now define the isomorphism for two database instances A and B over the one relation R. With regards to an isomorphism $h: A \longrightarrow B$, h is one-to-one. Moreover, for each relational attribute P of R, and each tuple $(t_1, t_2, \dots, t_n)$, it is obtained that $(t_1, t_2, \dots, t_n) \in P^A$ iff $h(t_1), h(t_2), \dots, h(t_n)$ belongs to P^B. Therefore, A and B isomorphic providing that there is a homomorphism from A to B. We can now infer from the above statements for two database instances that they are isomorphic if one could be attained from the another by changing the name of elements of its active domain on one-to-one [1].

Lemma: Let q be a Boolean conjunctive query, and A a database instance. The consequent statements are equal [26].

- $A \vDash q$
- There exists a homomorphism $h: d[q] \longrightarrow A$ in which $d[q]$ exemplifies the canonical database of q.

Suppose that $q = \exists a_1 \dots \exists a_n \, \phi(a_1, a_2, \dots, a_m)$. For the first expression in the lemma $A \vDash q$, there exists attributes $(b_1, b_2, \dots, b_m)$ in adom(A) which means $A \vDash \phi(b_1, b_2, \dots, b_m)$. Hence, the function h with $h(a_i) = b_i$ is a homomorphism $h: d[q] \longrightarrow A$. For the second expression in the lemma $h: d[q] \longrightarrow A$, the values $h(a_i) = b_i$ provide values for the explication of the $\exists a_i$ of q in the adom(A) such that we obtain $A \vDash \phi(b_1, b_2, \dots, b_m)$ [26].

Homomorphism Theorem: Based on the theorem, the consequent statements are equivalent for the conjunctive queries q_1 and q_2 [26] [1].

- $q_1 \subseteq q_2$
- There exists a homomorphism $h: d[q_2] \longrightarrow d[\, q_1]$ in which $d[q_i]$ exemplifies the canonical database of q_i.
- $d[\, q_1] \vDash q_2$

For the first expression $q_1 \subseteq q_2$, we can say $d[\, q_1] \vDash q_2$ because of $d[\, q_1] \vDash q_1$. Therefore, there exists a homomorphism $h: d[q_2] \longrightarrow d[\, q_1]$ according to the lemma. For the third expression $d[\, q_1] \vDash q_2$, there is a homomorphism from q_2 to q_1 by the lemma. And then, it should

be proved that if $A \vDash q_1$, then $A \vDash q_2$. At that case, $A \vDash q_1$ verifies there exists a homomorphism h': $d[\,q_1] \longrightarrow A$ [1] [26].

The complexity of the problem is obviously in NP. In order to demonstrate the problem is NP-Complete, Chandra and Merlin reduced the graph colouring problem. Let G is a graph and k is an integer. With respect to the queries q_1 and q_2, q_2 is a folding of q_1 if and only if the graph G could be k-coloured. Suppose that R is a single binary relation and V is the set of vertices such that,$E \subset V^2$ represents the symmetrical connection for the set of edges. Then, the following statements can be written for the queries q_1 and q_2 where K is the set of k variables as disjointed from V [1].

$$q_1 = (). \exists V. \exists K. (\bigwedge_{v,w \in E} R(v,w)) \wedge (\bigwedge_{\substack{v,w \in K \\ and\ v \neq w}} R(v,w))$$

$$q_2 = (). \exists K. \bigwedge_{\substack{v,w \in K \\ and\ v \neq w}} R(v,w)$$

Basically, the main point of the above statement is that there exists a minimal equal query for each conjunctive query, which might be attained by folding primary query. Additionally, folding a query cannot cause an enhancement in the number of variables or atomic formulae in the query. For this reason, the recent query is isomorphic to the primary one [1].

To show the reduction from 3-colorability, let q_G be the Boolean conjunctive query that is equal to the graph G. For instance, consider the graph consists of edges (k,l), (l,m), (m,n), (m,k). At that case, q_G: $-R(k,l), R(l,m), R(m,n), R(m,k)$. Assume that the query q_{K_3}: $-R(v,w), (w,z), (z,v)$. Apparently, G is 3-colorable iff there is a homomorphism from q_G to q_{K_3}. Likewise, G is 3-colorable iff $q_{K_3} \subseteq q_G$ in accordance with the homomorphism theorem [1].

Theorem – Sagiv and Yannakakis (1980): *The query containment problem for union of conjunctive queries is NP-complete.*

The following statements are equal for every conjunctive query $q = q_1 \cup q_2 \cup \ldots \cup q_K$, $q' = q_1' \cup q_2' \cup \ldots \cup q_L{}'$ [3] [2] [5].

- $q_1 \cup q_2 \cup \ldots \cup q_K \subseteq q_1' \cup q_2' \cup \ldots \cup q_L{}'$.
- For $i = 1, \ldots, K$ and $j = 1, \ldots, L$ $q_i \subseteq q_j{}'$

The demonstration of Sagiv and Yannakakis makes use of Chandra and Merlin theorem. Let DB be a database instance while t is a tuple. Suppose that $t \in q(DB)$, such that $t \in q_i(DB)$. We know that $q_i \subseteq q_j{}'$ with respect to $j, t \in q_j{}'(DB)$. Hence, it is straightforward that $t \in q'(DB)$.

Let $D[\,q_i]$ be canonical instance, so it is obtained that $q_i(D[\,q_i]) \subseteq q(D[\,q_i]) \subseteq q'(D[\,q_i])$. This provides $init(\,q_i) \in q_j{}'(D[\,q_i])$ where $init$ represents the initial variables of query, which is what homomorphism theorem requires that $q_i \subseteq q_j'$ [3] [2].

Sagiv and Yannakakis used the generalization of tableaux on conjunctive queries for union and difference such that tableau can be designed as a two-dimensional projection of queries, which is a

database that variables take part in tuples together with a group of selected variables which are shown in answer line in the below figure. To illustrate;

Let q be a conjunctive query $q(a,b) :- P(a,b), R(b,c), R(b,d), R(d,b)$, so the tableau of q

	X	Y			X	Y
P:	a	b		**R:**	b	c
					b	d
					d	b
	a	b ◄— Answer Line				

A homomorphism between two tableaux is a mapping that $f: T_1 variables \rightarrow T_2\ variables + constants$. At that case, for each selected variable a, $f(a) = a$. Besides, for each relation R in T_1 as well as line $(a_1, a_2, \dots, a_k)$, the tuple $(f(a_1), f(a_2), \dots, f(a_k))$ is a line of relation in T_2. To make it clearer, we can consider two queries on one relation [27].

$q_1(a,b) :- R(b,a), R(a,c)$

$q_2(a,b) :- R(b,a), R(d,a), R(a,e)$

Then we can obtain these two tableaus for the given queries.

	T q_1			**T q_2**	
	X	Y		X	Y
R:	b	a	**R:**	b	c
	a	c		d	a
				a	e
	a	b		a	b

From the tableaus, according to the mapping function f; $f(d) = b$, $f(e) = c$, $f(a) = a$ and $f(b) = b$. And therefore, $q_1 \subseteq\ q_2$.

Let $\vec{t}$, DB, $\vec{a}$ be a tuple, a database instance, and the tuple of selected variables respectively. $\vec{t}$ is in the answer to a conjunctive query q on DB if and only if there exists a homomorphism f to DB from the tableau of q, so that $f(\vec{a}) = \vec{t}$. Suppose that variables $v_1,\ v_2$ belong to conjunctive queries q_1 and q_2 in turn. Assume there is a homomorphism $f: \mathrm{T}\ q_1 \rightarrow \mathrm{T}\ q_2$, and $\vec{t} \in\ q_1(DB)$. Then, there exists a homomorphism k from T q_1 to DB, so $g(\vec{a}) = \vec{t}$. Consider the composition $h(.) = g(f(.))$ that is a homomorphism to DB from T q_2, so that $h(\vec{a}) = \vec{t}$. Therefore, $\vec{t} \in\ q_1(DB)$ [27].

In case $q_1(DB) \subseteq\ q_2(DB)$, let T $q_1 \rightarrow$ be database instance as DB. Apparently, $\vec{a}$ is in the answer to q_1 for T q_1. Hence, we attain $\vec{a} \in\ q_2(\mathrm{T}\ q_1)$. Consequently, we can say there exist a homomorphism from T q_2 to T q_1, so that $f(\vec{a}) = \vec{a}$ [27].

Theorem – Klug (1988): *The query containment problem for conjunctive queries with arithmetic operations is $\prod_2^p -$ complete.*

As Klug defined in 1988, conjunctive queries with arithmetic operations were not been examined until he has done, whereas, it was somewhat significant to take into consideration for containment, equivalence, and minimization. Klug's main objective was to find an effective algorithm to test containment.

To recall, the containment $q_1 \subseteq q_2$ for the perspective of the equality queries q_1 and q_2 has validity if and only if there exists a homomorphism to q_1 from q_2. The 'if' part is simply verified whilst demonstrating the 'only if' part necessitates q_1 to be a database itself, which is implemented by using one to one valuation p to q_1 (Klug describes valuation as a function from $x_q \cup y_q$ to q, where x_q is a set of selected variables and y_q is a set of non-selected (existentially quantified) variables). To make it clearer, $p(\ q_1\ )$ is the database instance D, the set of tuples D_R can be indicated as $p(z_1), p(z_2), \dots, p(z_n)$ for every conjunct $R(z_1, z_2, \dots, z_n)$ where R is the relation. In that case, $q_1(p(\ q_1)) \subseteq q_2(p(\ q_1))$ on condition that $q_1 \subseteq q_2$. When a_1 is the abstract of q_1, it is obtained that $p(\ a_1\ ) \in q_1(p(\ q_1))$, and therefore $p(\ a_1\ ) \in q_2(p(\ q_1))$ that becomes homomorphism from q_2 to q_1 by having composition with p^{-1} [8].

According to Klug, if they carried out abovementioned method for conjunctive queries with arithmetic operations, individual rational numbers would be assigned by the valuation p to the variables of q_1. Inasmuch as the set of rational numbers is in order, each pair of different values take part in no more than a relationship. In other words, irrespective of any arithmetic operation in the query, $p(x) < p(y)$ or $p(x) > p(y)$ if $p(x)$ is not equal to $p(y)$. These counterfeit operations give rise to appearance of $p(\ a_1\ )$ in $q_2(p(\ q_1))$ despite of the failure of the containment. As an example,

$$q_1 = \{k: (\exists_{l,m} R(k, l, m)\}$$

$$q_2 = \{k: (\exists_{l,m} R(k, l, m)\ \&\ l < m\}$$

$$p = \{k \to 0, l \to 1, m \to 2\}$$

Above statements show that $p(\ a_1\ ) \in q_2(p(\ q_1))$, however, $q_1 \subseteq q_2$ is out of question. Consequently, a single canonical database does not ensure the most appropriate test of containment for conjunctive queries with arithmetic operations. In Klug's method, the convenient procedure is to consider numerous (increase incrementally) *representative databases* instead of the single database $p(\ q_1)$, these representative databases stand for a dissimilar suitable order of variable assignments through the rational number line. His algorithms for test of containment are not in NP, membership is in $\prod_2 p$ [8].

Theorem – van der Meyden (1992): *The query containment problem for conjunctive queries with inequalities is $\prod_2^p -$ complete.*

The van der Meyden's theorem consists of the following results which are obtained by the reduction from 3-colorability [6].

- For conjunctive monodical {≠}-queries, there is a {<}-database DB that has NP-hard complexity.
- In monodical {≠} – databases, there is a consecutive query that has co-NP hard data complexity.

He specified two states, which are sequential queries and databases of bounded width, with PTIME combined complexity, and the theorem defined above enounces that none of the two states might be generalised to the queries and databases including the operation '≠'. Moreover, he adds that the confirmation of the theorem could be altered to has merely inequality, putting forward that conjunctive queries with inequalities and monodical {≠} – databases have $\prod_2^p - complete$ complexity. That is because the complexity of the problem is co-NP hard and NP-hard simultaneously [6].

THE COMPLEXITY of THE QUERY CONTAINMENT PROBLEM UNDER BAG (MULTISET) SEMANTICS

The query containment problem under set semantics has been largely sorted out as given above theorems and approaches. However, SQL queries have bag semantics which play tremendous role for real database management systems. In this chapter, theorems regarding the query containment problem under bag semantics.

Theorem - Chaudhuri and Vardi (1993): *The query containment problem under bag semantics for conjunctive queries is* $\prod_2^p - hard.$

The problem under bag semantics was put forward by Chaudhuri and Vardi in the first instance. In their paper, they investigated the optimization of real conjunctive queries, and ultimately defined that optimization techniques from the set semantics are not conveyed to the bag semantics.

Apart from the query containment under set semantics, a query contains another providing that the response to the posterior is every time a sub multiset of the response to the prior [10].

Chaudhuri and Vardi stated that the query containment problem under bag semantics is, of course, more difficult than the query containment problem under set semantics. First and foremost, they also defined that the certain complexity of the problem is waiting to be solved even though they came up with $\prod_2^p - hard.$ Besides, they appended that they did not know the problem is whether decidable or not.

Despite the major ambiguity in the bag containment, they studied equivalence of conjunctive queries such that equivalence is reducible to containment. According to their theorem, $q \equiv_b q'$ if and only if q and q' are isomorphic in which q and q' are conjunctive queries. Consequently, equivalence of conjunctive queries under bag semantics is equal to graph isomorphism as polynomial [10].

Theorem - Ioannidis and Ramakrishnan (1995): *The query containment problem under bag semantics for union of conjunctive queries is undecidable.*

Ioannidis and Ramakrishnan examined conjunctive queries with regards to databases including tuples with 'associated label' [28]. They built up conclusions on the decidability making use of

distinct 'label systems' which hold the relations as bags. When they proved their problem, a version of Diophantine equation problem (Hilbert's 10th problem) was used for reduction [4].

Suppose that $P(a_1, a_2, \dots, a_n)$ is a polynomial P consisting of integer coefficients. Diophantine equation corresponds to the statement $P = 0$ while just all-integer solutions are being looked for. Establishing the existence of positive integers solution to the given statement is an undecidable problem. To show it more technically, it is undecidable that whether $\varphi \leq \delta$ is accurate according to two polynomials $\varphi(a_1, a_2, \dots, a_n)$ and $\delta(a_1, a_2, \dots, a_n)$ containing nonnegative integer coefficients, as well as without any constant terms. Ioannidis and Ramakrishnan used this undecidable problem for their proof [4].

In the theorem, L is a label system which has the set {0, +, *, ≤, l} in which l represents the cluster of positive integers. Let q_1 and q_2 be two unions of conjunctive queries. As for L, $q_1 \leq_r q_2$ is undecidable where $q_1 = \bigcup_{i=1}^{k_1} x_i$ and $q_2 = \bigcup_{j=1}^{k_2} y_j$ [4].

Looking at the proof, two polynomials $\varphi(a_1, a_2, \dots, a_n)$ and $\delta(a_1, a_2, \dots, a_n)$ are considered with n variables just as defined above. Consider the φ as;
$\varphi(a_1, a_2, \dots, a_n) = \sum_{i=1}^{k} x_i a_1^{b_{i1}} a_2^{b_{i2}} a_3^{b_{i3}} \dots a_n^{b_{in}}$ (in which x, b are positive and nonnegative integers in turn). They tailor this statement to union of conjunctive queries using joins of unary relations in order to codify monomials that refers to products of every variable in the statement [5]. Let A represent the unary relation for every variable and P conjunctive query such that every conjunctive query has the below model (every relation participates in the identical domain D) [4].

$$P(u) = A_1(u) * A_1(u) * A_1(u) *\dots * A_1(u) *\dots* A_n(u) * A_n(u) * A_n(u) *\dots * A_n(u)$$

d_{i1} TIMES $\quad$ d_{in} TIMES

Subsequently, the following statement should be provided.

$\varphi(a_1, a_2, \dots, a_n) \leq \delta(a_1, a_2, \dots, a_n) \leftrightarrow \bigcup_{i=1}^{k_1} x_i \leq_r \bigcup_{j=1}^{k_2} y_j$ [4]

Let P_x and P_y represent $\bigcup_{i=1}^{k_1} x_i$ and $\bigcup_{j=1}^{k_2} y_j$ respectively. Therefore, the following equations are provided for every component $e \in D$.

$$P_x(e) = \varphi(A_1(e), A_2(e), A_3(e), \dots, A_n(e))$$
$$P_y(e) = \delta(A_1(e), A_2(e), A_3(e), \dots, A_n(e))$$

As a result, in case $\bigcup_{i=1}^{k_1} x_i \leq_r \bigcup_{j=1}^{k_2} y_j$, the statement $P_x(e) \leq P_y(e)$ is provided. Since the equation problem is undecidable, the query containment problem under bag semantics for union of conjunctive queries is undecidable either [4].

Theorem – Jayram, Kolaitis, Vee (2006): *The query containment problem under bag semantics for conjunctive queries with inequalities is undecidable*.

Jayram, Kolaitis and Vee utilized Hilbert's tenth problem for reduction so as to demonstrate their opinion. Looking at the proof, they define two polynomials of the same kind $\varphi(a_1, a_2, \dots, a_n)$ and $\delta(a_1, a_2, \dots, a_n)$ comprising integer coefficients. The relation between the polynomials

is $\varphi(a_1, a_2, \dots, a_n) \leq {a_1}^d \delta(a_1, a_2, \dots, a_n)$ in which d shows the degree of the polynomial. The aim of the theorem is to come up with two conjunctive queries q_1, q_2 with inequalities under multiset semantics for every given integer $(a_1, a_2, \dots, a_n)$ under the circumstance $q_1 \leq q_2$ [7] [5].

The demonstration of the theorem is implemented in three stages. In the first stage, they define two conjunctive queries *without inequalities*. After that, they circumscribe the queries on 'polynomial encoders' which is a particular form of database instances. By carrying out the rest of the method comprehensively, they denote that the query containment problem for conjunctive queries without any inequalities under bag semantics is undecidable providing limited to an exclusive format of database [5] [7].

In the second stage, they evaluate the conjunctive queries *with inequalities* for the purpose of attaining the next statements. Should a database instance I is of exclusive format, the former stage is considered. Should a database instance I is not of exclusive format, the $q_1 \leq q_2$ is indispensable. At that stage, the main objective is to have complete structure by using inequalities. In the third stage, they define that it is controversial whether containment problem with inequalities is decidable as long as conjunctive queries contain a single binary relation [5] [7].

Theorem: Afrati, Damigos, Gergatsoulis (2010): Afrati, Damigos, and Gergatsoulis specified the containment problem for some exclusive subclasses of conjunctive queries as decidable.

- *The query containment problem under bag semantics for conjunctive queries which do not include projections is O (n^2 log(n))* [29].

Let q_1 and q_2 be conjunctive queries devoid of projections. The equation $q_1 \leq q_2$ is valid if and only if there exists 'subgoals-onto' containment mapping to q_2 from q_1. Here, it should be controlled whether the mapping of selected variables is subgoals-onto at the same time. Putting in order the subgoals between two queries could be completed in O (n^2 log(n)) [29].

- *The query containment problem under bag semantics for conjunctive queries which do not include self-joins is O (n log(n))* [29].

Let q_1 be a conjunctive query when q_2 is a conjunctive query devoid of self-joins such that each relation name exists once at the furthest, and each subgoal belonging q_1 corresponds to a subgoal belonging q_1 once at the furthest. These procedures and sorting relation names can be done in *O (n log(n))* [29].

CONCLUSION

In this paper, the containment problem under set and bag semantics were examined. Generally accepted, the problem under set semantics was sorted out, and it also has been known as NP-complete. On the other hand, even though more than twenty years passed over its first appearance, the query containment problem under bag semantic retains open to date. The table including findings of the problem is shown below.

QUERIES	COMPLEXITY UNDER SET SEMANTICS	COMPLEXITY UNDER BAG SEMANTICS
Conjunctive Queries	NP-Complete (Chandra & Merlin – 1977)	OPEN
Unions of Conjunctive Queries	NP-Complete (Sagiv & Yannakakis – 1980)	Undecidable (Ioannidis & Ramakrishnan- 1995)
Conjunctive Queries with Arithmetic Operations	$\prod_2^p$ –Complete (Klug 1988)	
Conjunctive Queries with Inequalities	$\prod_2^p$ – Complete (van der Meyden -1992)	Undecidable (Jayram & Vee & Kolaitis-2006)
Conjunctive Queries without Projections		*O* (n^2 *log(n))*
Conjunctive Queries without Self-Joins		*O (n log(n))*

REFERENCES

[1] A. K. Chandra and P. M. Merlin, “Optimal implementation of conjunctive queries in relational data bases,” Boulder, Colorado, 1977.

[2] P. Koutris, “Query Containment,” 2016. [Online]. Available: http://pages.cs.wisc.edu/~paris/cs838-s16/lecture-notes/lecture2.pdf. [Accessed 2 August 2017].

[3] Y. Sagiv and M. Yannakakis, “Equivalences Among Relational Expressions with the Union and Difference Operators,” *Journal of the ACM (JACM),* vol. 27, no. 4, pp. 633-655 , 1980.

[4] Y. E. Ioannidis and R. Ramakrishnan, “Containment of conjunctive queries: beyond relations as sets,” *ACM Transactions on Database Systems (TODS),* vol. 20, no. 3, pp. 288-324, 1995.

[5] P. G. Kolaitis, “CEUR Workshop Proceedings,” 21 March 2013. [Online]. Available: http://ceur-ws.org/Vol-1087/keynote2slides.pdf. [Accessed 25 July 2017].

[6] R. v. d. Meyden, “The Complexity of Querying Indefinite Data about Linearly Ordered Domains,” *Journal of Computer and System Sciences,* vol. 54, no. 1, pp. 113-135, 1997.

[7] T. S. Jayram, P. G. Kolaitis and E. Vee, “The Containment Problem for Real Conjunctive Queries,” Chicago, 2006.

[8] A. Klug, “On conjunctive queries containing inequalities,” *Journal of the ACM (JACM),* vol. 35, no. 1, pp. 146-160, 1988.

[9] T. J. Green, “Bag Semantics,” in *Encyclopedia of Database Systems*, L. Liu and M. T. Ozsu, Eds., Springer US, 2009, pp. 201-206.

[10] S. Chaudhuri and M. Y. Vardi, “Optimization of real conjunctive queries,” Washington, 1993.

[11] O. Goldreich, Computational Complexity: A Conceptual Perspective, Cambridge University Press, 2008.

[12] S. Rudich and A. Wigderson, Computational Complexity Theory, American Mathematical Society, 2004.

[13] J. Hartmanis, Computational Complexity Theory, American Mathematical Society, 1989.

[14] D.-Z. Du and K.-I. Ko, Theory of Computational Complexity, Wiley-Interscience, 2000.

[15] J. E. Savage, Models of Computation: Exploring the Power of Computing, AddisonWesley Pub, 1997.

[16] S. Arora and B. Barak, Computational Complexity: A Modern Approach, Cambridge University Press, 2009.

[17] J. Erickson, *Algorithms and Models of Computation, Lecture Notes,* University of Illinois at Urbana–Champaign, 2014.

[18] P. Gacs and L. Lovasz, *Complexity of Algorithms, Lecture Notes,* Boston University; Yale University, 1999.

[19] N.-j. Kwak, "Cook-Levin Theorem Proof and Illustration," in *WSAC2010*, 2010.

[20] M. Sipser, Introduction to the Theory of Computation, Course Technology, 2012.

[21] M. T. Goodrich and R. Tamassia, Algorithm Design: Foundations, Analysis and Internet Examples, John Wiley & Sons, 2001.

[22] C. Saha, "Examples of NP-Complete Problems, Lecture Notes," 2015. [Online]. Available: http://drona.csa.iisc.ernet.in/~chandan/courses/complexity15/notes/lec3.pdf. [Accessed 20 June 2017].

[23] L. Mouatadid, "3-Colouring is NP-complete, Lecture Notes," 2014. [Online]. Available: http://cs.bme.hu/thalg/3sat-to-3col.pdf. [Accessed 5 July 2017].

[24] I. Potapov, "3-Coloring is NP-Complete, Lecture Notes," 2017. [Online]. Available: https://cgi.csc.liv.ac.uk/~igor/COMP309/3CP.pdf. [Accessed 5 July 2017].

[25] E. W. Weisstein, "MathWorld," 2017. [Online]. Available: http://mathworld.wolfram.com/Isomorphism.html. [Accessed 5 July 2017].

[26] P. G. Kolaitis, "Principles of Database Systems," 2010. [Online]. Available: https://classes.soe.ucsc.edu/cmps277/Winter10/Lectures/lect10-w10.pdf. [Accessed 1 July 2017].

[27] R. Pichler, "Conjunctive Queries," 2017. [Online]. Available: https://www.dbai.tuwien.ac.at/staff/pichler/dbt/slides/dbt06.pdf. [Accessed 10 August 2017].

[28] Y. E. Ioannidis and E. Wong, "Towards an algebraic theory of recursion," *Journal of the ACM (JACM),* vol. 38, no. 2, pp. 329-381, 1991.

[29] F. N.Afrati, M. Domigos and M. Gergatsoulis, "Query containment under bag and bag-set semantics," *Information Processing Letters,* vol. 110, no. 10, pp. 360-369, 2010.

[30] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, Introduction to Algorithms, Second Edition, The MIT Press, 2001.