Отметим, что полученные результаты согласуются с результатами работ Чернявского А.Д. [9] и могут быть использованы на макроуровне для прогнозирования уровня распространения информационного воздействия на замкнутые социальные группы.

ЛИТЕРАТУРА

1. Коротаев А. Теория социального взрыва [Електронний ресурс] : стенограмма передачи «Наука 2.0» – совместного проекта информационно-аналитического портала «Полит.ру» и радиостанции «Вести FM» // Доктор исторических наук, заведующий кафедрой современного Востока РГГУ, ведущий научный сотрудник Института востоковедения и Института Африки РАН Андрей Коротаев. http://radiovesti.ru/episode/show/episode_id/10327. – 09 апреля, 2011 г.
2. http://www.rbc.ua/rus/news/economic/kolichestvo–abonentov–sotovoy–svyazi–v–ukraine–za–i–kvartal–17052013174300/
3. http://itc.ua/tag/iks–consulting/
4. Рейнгольд Г. Умная толпа: новая социальная революция / Г. Рейнгольд. — М. : ФАИР ПРЕСС, 2006. — 416 с.
5. Kermack W. O. A Contribution to the Mathematical Theory of Epidemics / W. O. Kermack and A. G. McKendrick // Proc. R. Soc. Lond. A August 1, 1927 115 772 700–721; doi:10.1098/rspa.1927.0118 [http://rspa.royalsocietypublishing.org/ content/115/772/700.citation#related–urls].
6. Братусь А. С. Динамические системы и модели биологии / А. С. Братусь, А. С. Новожилов, А. П. Платонов. — М. : ФИЗМАТЛИТ, 2010. — 400 с.
7. http://en.wikipedia.org/wiki/Epidemic_model#The_SIR_model.
8. Бабенко К. И. Основы численного анализа / К. И. Бабенко. — М. : Наука, 1986. — 744 с.
9. Чернявский А. Д. Информационная модель «социального взрыва» / А. Д. Чернявский // Современные исследования социальных проблем (электронный научный журнал). — 2012. — №1(09). — С. 406-426.

УДК 004.658.2

ANSWERING CONJUNCTIVE QUERIES OVER A TEMPORALLY-ORDERED FINITE SEQUENCE OF ABOXES SHARING ONE TBOX

Keberle N. G., Ph.D., docent

Zaporizhzhya National University

Ontology-based data access (OBDA) assumes that data in a database are mediated with a conceptual layer, available for clients and hiding data storage details. Ontologies are good candidates for such a conceptual layer presentation, whereas databases are good for huge data storage.

The benefits from combination of databases and knowledge bases are as follows:

– database management is the mature field of research, there is a lot of commercially and freely available DBMSs, showing good-to-excellent performance on large datasets. It is an obvious place to store the assertional part of some knowledge base, i.e. an ABox;

– a TBox often requires a reasoning support to deduce additional assertions, axioms and to check the consistency of a knowledge base.

At the same time, employing such an approach is rather challenging due to significant differences between relational database systems and ontology languages, based on Description Logics, such as OWL. At first, relational databases adopt a closed-world semantics, i.e. all facts that are not explicitly stated to be true are assumed to be false. In contrast, OWL is based on an open world semantics which does not require one to fix the truth value of every fact and is more similar to an incomplete database. Second, relational databases are unaware of the intensional part of a knowledge base (called a TBox).

Research has been done so far in the OBDA field considers only one ABox stored in a data source, that is an actual set of assertions on individuals and their pairs. However, real applications show that ABox is changing over time. The examples of such dynamic systems can be easily found in practice: environmental conditions, air traffic load, computer system load and performance, health control for the people suffering from serious diseases. Therefore, in some applications of situation awareness, there is a need to store an archive of

ABoxes, keeping ABoxes actual at different time points. Temporal logics are often used as the means to formulate constraints a dynamic system should obey during its work.

The objective of the research presented in this paper is twofold: at first, to elaborate a language of unions of temporal conjunctive queries, which allows evaluating atemporal unions of conjunctive queries at different time points; and at second, to propose an algorithm of answering a union of temporal conjunctive queries, possessing termination, soundness and completeness properties.

Keywords: ontology-based data access, temporal conjunctive query language, description logic knowledge base.

Кеберле Н. Г. ВЫЧИСЛЕНИЕ ОТВЕТА НА КОНЪЮНКТИВНЫЙ ЗАПРОС НАД КОНЕЧНОЙ ПОСЛЕДОВАТЕЛЬНОСТЬЮ УПОРЯДОЧЕННЫХ ВО ВРЕМЕНИ ABOX, ИСПОЛЬЗУЮЩИХ ОДИН TBOX / Запорожский национальный университет, Украина

Доступ к данным на основании онтологий (OBDA) предполагает, что данные в базе данных сопровождаются некоторым концептуальным уровнем, который скрывает от клиентов детали хранения данных, и представляет их семантику. Онтологии являются хорошим кандидатом на реализацию этого концептуального уровня, тогда как системы управления базами данных подходят для хранения значительных объемов данных. Одним из интересных приложений OBDA является проверка конечного множества ограничений, заданных на некотором языке над упорядоченной во времени последовательностью баз фактов (ABox), использующих одну и ту же базу знаний (TBox), где каждое ограничение является конъюнктивным запросом. Предложен алгоритм ответа на конъюнктивный запрос в этом языке, доказаны конечность, непротиворечивость и полнота этого алгоритма.

Ключевые слова: доступ к данным на основании онтологий, язык темпоральных конъюнктивных запросов, база знаний в формализме логик дескрипций.

Кеберле Н. Г. ОБЧИСЛЕННЯ ВІДПОВІДІ НА КОН’ЮНКТИВНИЙ ЗАПИТ НАД СКІНЧЕННОЮ ПОСЛІДОВНІСТЮ ВПОРЯДКОВАНИХ ЗА ЧАСОМ ABOX, ЩО ВИКОРИСТОВУЮТЬ ОДИН TBOX / Запорізький національний університет, Україна

Доступ до даних на основі онтологій (OBDA) передбачає, що дані в базі даних супроводжуються деяким концептуальним рівнем, який приховує від клієнтів деталі збереження даних, та представляє їх семантику. Онтології є гарним кандидатом на реалізацію такого концептуального рівня, тоді як системи управління базами даних підходять для збереження значних об’ємів даних. Одним з цікавих застосувань OBDA є перевірка скінченної множини обмежень, заданих на певній мові над впорядкованою за часом послідовністю баз фактів (ABox), що використовують одну базу знань (TBox), де кожне обмеження є кон’юнктивним запитом. Запропонований алгоритм обчислення відповіді на кон’юнктивний запит у цій мові, доведені скінченність, несуперечливість та повнота цього алгоритму.

Ключові слова: доступ до даних на основі онтологій, мова темпоральних кон’юнктивних запитів, база знань у формалізмі логік дескрипцій.

INTRODUCTION

Ontology-based data access (OBDA) [1] assumes that data in a database are mediated with a conceptual layer, available for clients and hiding data storage details. Ontologies are good candidates for such a conceptual layer presentation, whereas databases are good for huge data storage.

The benefits from combination of databases and knowledge bases are as follows:

– database management is the mature field of research, there is a lot of commercially and freely available DBMSs, showing good-to-excellent performance on large datasets. It is an obvious place to store the assertional part of some knowledge base, i.e. an ABox;

– a TBox often requires a reasoning support to deduce additional assertions, axioms and to check the consistency of a knowledge base.

At the same time, employing such an approach is rather challenging due to significant differences between relational database systems and ontology languages, based on Description Logics, such as OWL. At first, relational databases adopt a closed-world semantics, i.e. all facts that are not explicitly stated to be true are assumed to be false. In contrast, OWL is based on an open world semantics which does not require one to fix the truth value of every fact and is more similar to an incomplete database. Second, relational databases are unaware of the intensional part of a knowledge base (called a TBox).

Research has been done so far in the OBDA field considers only one ABox stored in a data source, that is an actual set of assertions on individuals and their pairs. However, real applications show that ABox is changing over time. The examples of such dynamic systems can be easily found in practice: environmental conditions, air traffic load, computer system load and performance, health control for the people suffering from serious diseases. Therefore, in some applications of situation awareness [2], there is a need to store an archive of ABoxes, keeping ABoxes actual at different time points. Temporal logics are often used as the means to formulate constraints a dynamic system should obey during its work.

The objective of the research presented in this paper is twofold: at first, to elaborate a language of unions of temporal conjunctive queries, which allows evaluating atemporal unions of conjunctive queries at different time

points; and at second, to propose an algorithm of answering a union of temporal conjunctive queries, possessing termination, soundness and completeness properties.

The paper is organized as follows: in the next section a language of unions of temporal conjunctive queries is introduced and the definitions of main reasoning tasks available for such a query language are presented. In the section 3 the algorithm of answering temporal unions of conjunctive queries is presented and illustrated in examples. The section 4 is dedicated to the proofs of logical properties of the algorithm. The section 5 discusses the related work in the field of conjunctive queries answering.

RESEARCH ON CONJUNCTIVE QUERIES

Conjunctive queries over a knowledge base described in the description logic formalism [3] were introduced in [4] and investigated further in [5-7].

Assume there is a knowledge base K = (A, T), where T is a set of concept axioms (a TBox), A is a set of assertional axioms (an ABox). Fix a language of a knowledge base to ALC [3]. An interpretation of K, named *I*, is a pair ($\bullet I$, Δ), where Δ is a domain of individuals, obeying unique name assumption (UNA) and $\bullet I$ is an interpretation function, which assigns every concept C a set $C^I \subseteq \Delta$, every atomic role R a binary relation $R^I \subseteq \Delta \times \Delta$ and every individual name a an individual $a^I \in \Delta$. Assertional axioms are $C(a)$ – concept assertion and $R(a, b)$ – role assertion.

Query answering is the extension of a well known task of *instance checking*: given a knowledge base K and an assertion α. Check whether this assertion is entailed by an ABox of K.

Let Vars(Q) be a set of all distinguished and non-distinguished variables which appear in a query Q, let Inds(Q) to denote the set of all individual names which appear in query Q and Terms(Q) to denote the set of all terms in Q, i.e. Vars(Q) $\cup$ Inds(Q).

Let us formally define conjunctive queries and Boolean conjunctive queries for a well-elaborated language ALC [3].

Definition 1 (Conjunctive query, Union of conjunctive queries). Let x, y, c are respectively tuples of distinguished variables (answer variables), of non-distinguished variables and of individual names, and t, t_1, t_2 are terms in Terms(Q). A **conjunctive query (CQ)** is an expression of the form

$$conj(x, c) = \exists y.q_1 \wedge \ldots \wedge q_m,$$

where

$$q_i ::= C(t) \mid r(t_1, t_2).$$

A *Boolean conjunctive query* is a CQ without answer variables.

A *union of conjunctive queries* (*UCQ*) is a disjunction of conjunctive queries (CQs) of the form

$$Q(x) = \{x \mid conj_1(x, c) \vee \ldots \vee conj_n(x, c)\}.$$

Example 1. The example of a query asking about all students that attend some courses and take some exams could be as follows:

$$Q(x) = \{x \mid \exists y.takeCourse(x, y) \wedge takeExam(x, y)\}.$$

This query can be modified to a Boolean query by substitution of x with an individual name:

$$Q(x) = \{x \mid \exists y.takeCourse(\text{“}Eldora\text{”}, y) \wedge takeExam(\text{“}Eldora\text{”}, y)\}.$$

We use |Q| to denote the *size* of Q – the number of symbols required to build the query. The *arity* of a query will be the number of answer variables in the query. If all terms in Q are individual names, we say Q is *ground*. We write Q(c) for a query whose answer variables x are substituted by c, Q(x) for a conjunctive query and simply Q for a Boolean conjunctive query. Sometimes we write $x_1,\ldots, x_n$ instead of x, and similarly for y and c.

Given an ALC-knowledge base K = (T, A), an interpretation *I satisfies* a query Q(x) iff the interpretation function can be extended to the variables in Q(x) in such a way that *I* satisfies every term in Q(x). A query Q(x) is true w.r.t. K (written K |= Q) iff every interpretation that satisfies K also satisfies Q.

Definition 2 (Query answering, query entailment). Given a query Q(x) with a tuple of answer variables x, and a knowledge base K, a tuple of individuals c with the same arity of x is *an answer* for Q in K if $I \models Q(c)$ for every model *I* in K.

Given a Boolean conjunctive query Q, and a KB K, *query entailment* is a task to decide whether K |= Q if $I \models Q$ for every model *I* of K.

Given a conjunctive query Q(x), a tuple of individuals a, and a KB K, *query answering* is a task to decide whether a is an answer for Q(x) in K.

Known are several algorithms for answering unions of conjunctive queries over knowledge bases with static TBox and ABox, for example works of Ortiz [5], Glimm [6], Tessaris [4], Motik [7] should be mentioned. Any of those algorithms could serve as a basis for finding answers to atemporal CQs at particular time points, whereas possible extensions of those algorithms for the application to a sequence of ABoxes is an open question.

However, the aforementioned works are either focused on static ABox and TBox environment, thus having limited applicability in an evolving environment, or temporal languages proposed are unidirectional, and do not allow for queries asking both future and past time moments.

A language of temporal conjunctive queries with negation, together with the research on its computational and combined computational complexity is introduced in [8].

Transition graphs for a temporal query language answering over a finite set of versions of a database were investigated in [9]. The expressivity of a temporal query language presented is however restricted either to past [10], or to future [9], [11] direction of time.

TEMPORAL CONJUNCTIVE QUERIES

Let $K = (T, (A_i)_{0 \le i \le n})$ be a knowledge base with a sequence of ABoxes sharing one TBox. Let's describe a query language *extending* the language of conjunctions of positive existential formulae built from query atoms. Having in mind linear temporal logic LTL (see e.g. [12]), this language allows for the following temporal operators: O (next), O^- (previous), *U* (until), *S* (since).

Definition 3. Temporal conjunctive query (TCQ) is an expression

$$tconj(x, c) = \exists y.q_1 \wedge \ldots \wedge q_m,$$

where

$$q_i ::= \varphi \mid \Psi,$$

$$\varphi ::= C(t) \mid r(t_1, t_2),$$

$$\Psi ::= \varphi \mid \Psi_1 \vee \Psi_2 \mid \mathrm{O}\Psi \mid \mathrm{O}^-\Psi \mid \Psi_1 S \Psi_2 \mid \Psi_1 U \Psi_2$$

and *C* is a concept description, *r* is a role name, t, t_1, t_2 are terms in Terms(Q).

Derived temporal modalities like $\Diamond^-$(sometimes in the past), $\Box^-$ (always in the past), $\Diamond$, $\Box$ can be defined in a usual way (see, e.g. [12]).

Example 2. A query asking about students who had defended their thesis some time ago and had been ex-matriculated since then is expressed as follows:

$$Q(x) = \{x \mid \exists y.\ \Diamond^- Student(x) \wedge exMatriculated(x)\ S\ hasDefended(x, y)\}.$$

The semantics of the TCQ is defined as follows.

Definition 4. A total function $\pi : \mathrm{Terms}(\Psi) \to \Delta$ is a *binding* for a query Ψ in an interpretation *I*, if $\pi(a) = a$ for all individuals $a \in \mathrm{dom}(\pi)$, and the *validity* $I, \pi \models \Phi$ for atemporal CQ φ is defined inductively:

$$I, \pi \models C(t) \quad \text{iff} \quad I \models C(\pi(t)),$$

$$I, \pi \models r(t_1, t_2) \quad \text{iff} \quad I \models r(\pi(t_1), \pi(t_2)),$$

$$I, \pi \models \varphi_1 \wedge \varphi_2 \quad \text{iff} \quad I \models \varphi_1 \text{ and } I \models \varphi_2,$$

$$I, \pi \models \exists y \varphi \quad \text{iff} \quad \exists e \in \Delta: \pi' = \pi[y/e] \text{ and } I, \pi' \models \varphi$$

where the notation $\pi[y/e]$ represents a binding π extended with $\pi(y) = e$ if *y* is not in the domain of π, otherwise the original value for *y* is replaced by *e*.

The *validity* for a TCQ Ψ and a KB $K = (T, (A_i)_{0 \le i \le n})$ is extended as follows:

$$K, i, \pi \models \varphi \quad \text{iff} \quad \forall I \models_T A_i.\ I, \pi \models \varphi,$$

$$K, i, \pi \models \Psi_1 \wedge \Psi_2 \quad \text{iff} \quad K, i, \pi \models \Psi_1 \text{ and } K, i, \pi \models \Psi_2,$$

$$K, i, \pi \models \mathrm{O}\Psi \quad \text{iff} \quad i < n \text{ and } K, i+1, \pi \models \Psi,$$

$$K, i, \pi \models \mathrm{O}^-\Psi \quad \text{iff} \quad i > 0 \text{ and } K, i-1, \pi \models \Psi,$$

$$K, i, \pi \models \Psi_1 U \Psi_2 \quad \text{iff} \quad \exists k, i \le k \le n : K, k; \pi \models \Psi_2 \text{ and } \forall j, i \le j < k : K, j, \pi \models \Psi_1,$$

$$K, i, \pi \models \Psi_1 S \Psi_2 \quad \text{iff} \quad \exists k, 0 \le k \le I : K, k; \pi \models \Psi_2 \text{ and } \forall j, k \le j < i : K, j, \pi \models \Psi_1.$$

For a binding π, if, for every i, $\forall\ I \models_T A_i$. I, $\pi \models K$, this implies $I \models \Psi$. If such evaluation exists, we write $K \models \Psi$ and we say π is a match for Ψ in K. For a tuple of individuals $c_1,..., c_n$ mapped to a tuple of answer variables $x_1,\ldots,x_n$ we say $c_1,..., c_n$ is a *certain answer* for Ψ in K, iff $K \models \Psi[x_1,\ldots,x_n / c_1,..., c_n]$. We denote a set of certain answers for Ψ as Ans(Ψ).

Definition 5. A *union of temporal conjunctive queries* (UTCQ) Q(x) is a disjunction of temporal conjunctive queries (see Definition 3):

$$Q(x) = \{x \mid tconj_1(x, c) \vee \ldots \vee tconj_n(x, c)\}.$$

ANSWERING A UNION OF TEMPORAL CONJUNCTIVE QUERIES OVER A SEQUENCE OF ABOXES

Algorithm answering a union of temporal conjunctive queries

The idea of answering a UTCQ against a set of ABoxes is to use temporal operators as the means of detection of time points at which atemporal CQs should be evaluated. Due to the recursive nature of such temporal operators as S, U we have to store all the ABoxes and the values of particular CQs depending on the operator. Intuitively, given $\Psi = O\phi$ at a time point i, ϕ is evaluated at the time $i+1$, and so on.

To be able to combine certain answers obtained from different CQs of one TCQ, let's take a closer look at the nature of certain answers.

A certain answer to a CQ ϕ is a binding π of each $x_i \in x$ (distinguished variables) to some individual name that appeared in the KB K, such that in all models of K, $K \models \phi(\pi(x))$. There could be more than one certain answer for a CQ ϕ, so further we shall consider a set of certain answers for a query ϕ(x). A correspondent set of matches for ϕ actually produces some k-ary relation, where k is the arity of a CQ ϕ.

A certain answer to a UCQ Φ is a combination of answers of CQs in Φ, i.e. $c_1 \cup ... \cup c_n$ where n is a number of CQs in Φ. For such a combination there are two possible situations: (i) disjuncts ϕ_{j1}, ϕ_{j2} in UCQ Φ use pair wise disjoint sets of distinguished variables (i.e. there are no common distinguished variables in two arbitrary disjuncts of Φ); (ii) some disjuncts can share (some) distinguished variables of each other. To deal with sets of certain answers (that are actually relations) we adopt two operators of relational algebra, namely, $\times$ – a cross-product, and $\rhd\lhd$ – a natural join.

Cross-product operator $\times$ [13] is used for the case (i).

Definition 6. Given two bindings $\pi_1 : (x_1, \ldots, x_n) \to \Delta$, $\pi_1 : (y_1, \ldots, y_m) \to \Delta$, their cross-product, $\pi_1 \times \pi_2$ is a binding $\pi : X \to \Delta$ where x, y are free variables that do not have any variables in common, and $X = (x_1, \ldots, x_n, y_1, \ldots, y_m)$.

Join operator $\rhd\lhd$ [13] is used for the case (ii) to join two bindings w.r.t. common variables in both bindings are mapped to same constant.

Definition 7. Given two bindings $\pi_1 : (x_1, \ldots, x_n, z) \to \Delta$, $\pi_1 : (y_1, \ldots, y_m, z) \to \Delta$, their join, $\pi_1 \rhd\lhd \pi_2$ is a binding $\pi : X \to \Delta$ where x, y, z are free variables and $X = (x_1, \ldots, x_n, y_1, \ldots, y_m, z)$, iff every common variable z must be mapped to same constant $c \in \Delta$.

A correspondent binding for Φ will be: for (i) $\pi = \pi_{\phi 1} \times \ldots \times \pi_{\phi n}$, and for (ii) $\pi = \pi_{\phi 1} \rhd\lhd \ldots \rhd\lhd \pi_{\phi n}$.

The following theorems show applications of $\times$ and $\rhd\lhd$ for bindings.

Theorem 1. Given a formula $\Phi = \phi_1 \wedge \phi_2$, where ϕ_1, ϕ_2 are CQ formulas, a binding $\pi = \pi_1 \rhd\lhd \pi_2$ is a match for Φ iff bindings π_1, π_2 are matches for π_1, π_2.

Proof. It is true based on the definition of the join operator.

Theorem 2. Given a formula $\Phi = \phi_1 \vee \phi_2$, where ϕ_1, ϕ_2 are CQ formulas, a binding $\pi = \pi_1 \times \pi_2$ is a match for Φ iff the binding π_1 is a match for ϕ_1 or the binding π_2 is a match for ϕ_2.

Proof. The $\Rightarrow$ direction is trivial.

For $\Leftarrow$ direction, assume $\pi_1 : (x_1, \ldots, x_n, z) \to \Delta$, $\pi_1 : (y_1, \ldots, y_m, z) \to \Delta$, and they are matches for ϕ_1 and ϕ_2. From the nature of disjunction, we know that formula Φ is satisfiable if either ϕ_1 or ϕ_2 is satisfiable. That means there is a match for either ϕ_1 or ϕ_2. If z appears in both of the CQs, renaming z in one of the CQs does not change the validity. Therefore, we have that $\pi : (x_1, \ldots, x_n, z, y_1, \ldots, y_m, z') \to \Delta$ which is obtained from $\pi_1 \times \pi_2$ is indeed a match for Φ.

Now, consider a structure of a certain answer to a union of temporal CQs (UTCQ). It is a combination of answers to a (set of) TCQ obtained at proper time points, referred by temporal operators used in a UTCQ.

One more thing to be explicitly addressed is that known algorithms for conjunctive query answering, such as [6],[7], are focused on query entailment, that is, a Boolean conjunctive query answering. This means that the task of answering an atemporal CQ requires a preprocessing step, and considers a Boolean conjunctive query answering algorithm as a black box. Namely, at the preprocessing step a candidate match (a tuple of variables, substituted via some binding π with a tuple of individuals c) is submitted to a Boolean conjunctive query answering engine, and that engine decides if such a candidate match is a certain answer.

Now, present the algorithm informally.

Eliminate temporal operators in a UTCQ. The important step in our algorithm is to get a normal form where the temporal operators are used to decide at which time point each CQ should be evaluated. This is done by iterative application of the expansion rules Table 1. For every O and O^- operator, we just shift one point forward and backward. By doing these, we obtain a query that is in normal form whose atoms are UCQs, except some recursion atom which is a TCQ.

Replace the Boolean operators with relational operators. Every conjunction is replaced with join and every disjunction – with cross-product.

Retrieve an answer. Use an arbitrary query answering algorithm [4-7] as a black-box approach to compute a set of answers for a given UCQ. If the original UTCQ contains U, S, $\Box^-$, , $\Diamond^-$, $\Diamond$, the normal form of the transformed query might contain a recursion. In such case, if the time point $i < 0$ or $i > n$, then return Ø, else evaluate CQs with leading O or O^- for U, S and for derived modalities (if any).

Algorithm 1. Decide Q

Input:

K = (T, $(A_i)_{0 \le i \le n}$) : a knowledge base, consists of a TBox and a sequence of ABoxes at a time point i, $0 \le i \le n$

Q : a UTCQ

Output: Ans(Q, i) - a set of certain answers to Q at time point i

Ans(Q, i) = Ø

repeat

Ans′ = Ans(Q, i)

if Q = $TCQ_1 \vee TCQ_2$ then

Ans(Q, i) = Ans(TCQ_1, i) × Ans(TCQ_2, i)

end if

if Q = $TCQ_1 \wedge TCQ_2$ then

Ans(Q, i) = Ans(TCQ_1, i) $\rhd\lhd$ Ans(TCQ_2, i)

end if

if Q = O^- TCQ then

if i=1 then

Ans(Q, i) = Ø

else

Ans(Q, i) = Ans(TCQ, i - 1)

end if

end if

if Q = O TCQ then

if i=n then

Ans(Q, i) = Ø

else

Ans(Q, i) = Ans(TCQ, i + 1)

end if

end if

if Q = $TCQ_1\ U\ TCQ_2$ then

if $i=n$ then

$Ans(Q, i) = Ans(TCQ_2, i)$

else

$Ans(Q, i) = Ans(TCQ_2, i) \times (Ans(TCQ_1, i) \triangleright\triangleleft Ans(Q, i + 1))$

end if

end if

if Q = $TCQ_1\ S\ TCQ_2$ then

if $i=1$ then

$Ans(Q, i) = Ans(TCQ_2, i)$

else

$Ans(Q, i) = Ans(TCQ_2, i) \times (Ans(TCQ_1, i) \triangleright\triangleleft Ans(Q, i - 1))$

end if

end if

until Ans′ = Ans(Q, *i*)

return Ans(Q, *i*)

In Table 1, presented are some equivalence rules in LTL, used in Algorithm 1.

Table 1 – Equivalence rules of LTL for future operators. Taken from [12]

idempotent rule	$\Box\Psi \equiv \Box\Box\Psi,\ \Diamond\Psi \equiv \Diamond\Diamond\Psi,\quad \Psi_1 U (\Psi_1 U \Psi_2) \equiv \Psi_1 U \Psi_2,\quad (\Psi_1 U \Psi_2) U \Psi_2 \equiv \Psi_1 U \Psi_2$
commutativity rule	$\Box O\Psi \equiv O\Box\Psi,\quad \Diamond O\Psi \equiv O\Diamond\Psi,\quad O(\Psi_1 U \Psi_2) \equiv O\Psi_1 U O\Psi_2$
distributivity rule	$\Box(\Psi_1 \wedge \Psi_2) \equiv \Box\Psi_1 \wedge \Box\Psi_2, \Diamond(\Psi_1 \vee \Psi_2) \equiv \Diamond\Psi_1 \vee \Diamond\Psi_2,\quad O(\Psi_1 \wedge \Psi_2) \equiv O\Psi_1 \wedge O\Psi_2$ $O(\Psi_1 \vee \Psi_2) \equiv O\Psi_1 \vee O\Psi_2,\quad (\Psi_1 \wedge \Psi_2) U \Psi_3 \equiv (\Psi_1 U \Psi_3) \wedge (\Psi_2 U \Psi_3),$ $\Psi_1 U (\Psi_2 \vee \Psi_3) \equiv (\Psi_1 U \Psi_2) \vee (\Psi_1 U \Psi_3)$
temporal recursion rule	$\Box\Psi \equiv \Psi \wedge O\Box\Psi,\quad \Diamond\Psi \equiv \Psi \vee O\Diamond\Psi,\quad \Psi_1 U \Psi_2 \equiv \Psi_2 \vee O(\Psi_1 \wedge \Psi_2)$
absorption rule	$\Diamond\Box\Diamond\Psi \equiv \Box\Diamond\Psi,\quad \Box\Diamond\Box\Psi \equiv \Diamond\Box\Psi$

For the illustration of Algorithm 1 consider some examples, assuming that Algorithm 1 returns a set Ans of answers to Ψ at the time point *i*.

Example 3. Given a TCQ query $\Psi = O^-(\Phi_1 U \Phi_2)$ at a point *i*.

$$Ans(\Psi, i) = Ans(O^-(\Phi_1 U \Phi_2))$$

$$= Ans((\Phi_1 U \Phi_2), i\text{-}1) \qquad /*\text{move back one point by } O^-$$

$$= Ans(\Phi_2 \vee (\Phi_1 \wedge O\Psi), i\text{-}1) \qquad /*\text{ expansion rule for } U$$

$$= Ans(\Phi_2, i\text{-}1) \times Ans(\Phi_1 \wedge O\Psi, i\text{-}1) \qquad /*\text{ transforming } \vee$$

$$= Ans(\Phi_2, i\text{-}1) \times (Ans(\Phi_1, i\text{-}1) \triangleright\triangleleft Ans(O\Psi, i\text{-}1)) \qquad /*\text{ transforming } \wedge$$

$$= Ans(\Phi_2, i\text{-}1) \times (Ans(\Phi_1, i\text{-}1) \triangleright\triangleleft Ans(\Psi, i)) \qquad /*\text{ move forward one point by O}$$

If $i = 0$ in $Ans(\Phi_2, i\text{-}1)$ and $Ans(\Phi_1, i\text{-}1)$, then the evaluation of Ψ is the empty set.

A more complex example is given below.

Example 4. Given a TCQ query $\Psi = \Diamond^-(\Phi_1 U \Phi_2)$ at a point *i*.

$$Ans(\Psi, i) = Ans(\Diamond^-(\Phi_1 U \Phi_2))$$

$$= Ans((\Phi_1 U \Phi_2) \vee O^-\Psi, i) \quad /*\text{ expansion rule for } \Diamond^-$$

$$= \text{Ans}(\Phi_1 \, U \, \Phi_2, i) \times \text{Ans}(O^- \Psi, i) \text{ /* transforming } \vee$$

$$= \text{Ans}(\Phi_1 \, U \, \Phi_2, i) \times \text{Ans}(\Psi, i\text{-}1) \quad \text{/* move back one point by } O^-$$

$$= \text{Ans}(\Phi_2 \vee (\Phi_1 \wedge O\Psi'), i) \times \text{Ans}(\Psi, i\text{-}1) \text{ /* We substitute } \Phi_1 \, U \, \Phi_2 \text{ with } \Psi' \text{ and apply the expansion rule for } U$$

$$= \text{Ans}(\Phi_2, i) \times \text{Ans}(\Phi_1 \wedge O\Psi', i) \times \text{Ans}(\Psi, i\text{-}1) \quad \text{/* transforming } \vee$$

$$= \text{Ans}(\Phi_2, i) \times (\text{Ans}(\Phi_1, i) \rhd\lhd \text{Ans}(O\Psi', i)) \times \text{Ans}(\Psi, i\text{-}1) \quad \text{/* transforming } \wedge$$

$$= \text{Ans}(\Phi_2, i) \times (\text{Ans}(\Phi_1, i) \rhd\lhd \text{Ans}(\Psi', i\text{+}1)) \times \text{Ans}(\Psi, i\text{-}1) \quad \text{/* move forward one point by O}$$

If $i = n$ in $\text{Ans}(\Psi, i+1)$, then $\text{Ans}(\Psi, i+1)$ is evaluated to the empty set.

There is one thing we have to ensure that in the intersection of two sets of answers for conjunction of CQs a certain answer is obtained, i.e. there is a common answer for both CQs, otherwise an empty set. One way to do this is to retrieve all answers for each CQ and then to intersect them to get some common answers. Another way is first to retrieve an answer of a UCQ and then to decide if this answer is also the answer for the other CQs in the conjunction, otherwise keep retrieving and deciding until there is no more answer obtained. The former way is preferred since it offers more practical solution. It means that we can deal with it using relational algebra operators or database language operators.

Termination, soundness, completeness of the algorithm

Definition 8. (UTCQ closure). Given a temporal union of conjunctive queries Q, its *closure set*, *Cl*(Q) is a set of query atoms closed under the following rules

$$\text{if } q \in \text{Q then } q \in Cl(\text{Q}),$$

$$\text{if } O^- q \in \text{Q then } q \in Cl(\text{Q}),$$

$$\text{if } O\, q \in \text{Q then } q \in Cl(\text{Q}),$$

$$\text{if } q_1 \wedge q_2 \text{ then } q_1, q_2 \in Cl(\text{Q}),$$

$$\text{if } q_1 \vee q_2 \text{ then } q_1, q_2 \in Cl(\text{Q}),$$

$$\text{if } q_1 \, U \, q_2 \text{ then } q_1, q_2, O(q_1 \, U \, q_2) \in Cl(\text{Q}),$$

$$\text{if } q_1 \, S \, q_2 \text{ then } q_1, q_2, O^-(q_1 \, S \, q_2) \in Cl(\text{Q}).$$

Since a closure set for a UTCQ is finite, Algorithm 1 terminates after a finite number of steps.

Theorem 3. (Local) termination. Given a UTCQ Q and a knowledge base $K = (T, (A_i)_{0 \le i \le n})$. Algorithm 1 always terminates.

Proof. We can show the local termination inductively.

Base case. Any query is also contained in the closure set of itself.

Inductive case.

$C(a)$, $r(a_1, a_2)$ If we have a query Q which is atomic, then the closure set contains $C(a)$ or $r(a_1, a_2)$.

(O^- TCQ) For such query $Cl(\text{Q}) = \{\text{TCQ}, O^- \text{TCQ}\}$, i.e. evaluated are two elements, and in case of $i = 0$ the value of O^- TCQ is known to be Ø, so Algorithm 1 stops after two evaluations.

($\text{TCQ}_1 \, U \, \text{TCQ}_2$) For such query $Cl(\text{Q}) = \{\text{TCQ}_2, \text{TCQ}_1, \text{TCQ}_1 \, U \, \text{TCQ}_2, O(\text{TCQ}_1 \, U \, \text{TCQ}_2)\}$.

($\text{TCQ}_1 \, S \, \text{TCQ}_2$) For such query $Cl(\text{Q}) = \{\text{TCQ}_2, \text{TCQ}_1, \text{TCQ}_1 \, S \, \text{TCQ}_2, O(\text{TCQ}_1 \, S \, \text{TCQ2})\}$.

Theorem 4. Soundness. If for UTCQ Q its answer set $\text{Ans}(\text{Q}(x), i)$, obtained with Algorithm 1, is not empty, then Q has at least those certain answers that are in $\text{Ans}(\text{Q}, i)$.

Proof. We prove by induction. We start with evaluating non-temporal query, i.e. a query containing no temporal operator.

Base case. If we have an atomic query in the form of $C(a)$, then using any approach of CQ answering we obtain all the answers for the query Q entailment over $K = (T, (A_i)_{0 \le i \le n})$. If $K \models C(a)$ and $a \in \text{Ans}(\text{Q}(x), i)$, the function returns a and this value is stored in *Ans(Q(x); i)*. By Definition 4, this result tells us that the individual a is a certain answer to the query $C(x)$ w.r.t. the match $\pi(x) = a$. The same result is obtained if we have atomic query in the form of $r(a, b)$.

Inductive case can be obtained by Definition 4.

Theorem 5. Completeness. If a UTCQ Q has a certain answer *ans*, then Algorithm 1 shows that this answer is in Ans(Q, *i*).

Proof. By contradiction. Assume that (i) Q(*x*) has a certain answer *ans* w.r.t π, (ii) Ans(Q, *i*) - is a set of certain answers obtained by Algorithm 1, and (iii) *ans* ∉ Ans(Q, *i*). By (i), we know that K |= Q(*ans*) and that for all time points $0 \le i \le n$ in all models *I*, such that *I* |= K, *I* |= Q(*ans*). By (ii), for Algorithm 1 to return Ans(Q, *i*) such that *ans* ∉ Ans(Q, *i*) there are several reasons for it.

Q is atomic. If Q is atomic, i.e. in the form *C*(*x*) or *r*(*x*, *y*), then we know that Ans(Q, *i*) does not contain ans. This means that there is a model *I* of a knowledge base K which does not entail Q(*ans*). But this is a contradiction to our assumption (i).

($TCQ_1 \wedge TCQ_2$). If Ans(Q, *i*) does not contain *ans*, according to Algorithm 1 it means that *ans* ∉ Ans(TCQ_1, *i*) ⊳⊲ Ans(TCQ_2, *i*). This, in turn, leads to the existence of a model *I* of a knowledge base K such that *I* |= TCQ_1(*ans*) and *I* |≠ TCQ_2(*ans*) or vice versa, that contradicts to (i).

($TCQ_1 \vee TCQ_2$). If Ans(Q, *i*) does not contain *ans*, according to Algorithm 1 it means that *ans* ∉ Ans(TCQ_1, *i*) × Ans(TCQ_2, *i*). This, in turn, leads to the existence of a model *I* of a knowledge base K such that either *I* |≠ TCQ_1(*ans*) or *I* |≠ TCQ_2(*ans*), that contradicts to (i).

(O^- TCQ). If Ans(Q, *i*) does not contain *ans*, according to Algorithm 1 it means that *ans* ∉ Ans(Q, *i-1*). This, in turn, leads to the existence of a model *I* of a knowledge base K such that *I*, *i-1*|≠ Q(*ans*), that contradicts to (i).

(TCQ_1 *U* TCQ_2). If Ans(Q, *i*) does not contain *ans*, according to Algorithm 1 it means that *ans* ∉ Ans(TCQ_2, *i*) × (Ans(TCQ_1, *i*) ⊳⊲ Ans(Q, *i+1*)). This, in turn, leads to the existence of a model *I* of a knowledge base K such that either *I*, *i* |≠TCQ_2(*ans*) or *I*, *i* |≠TCQ_1(*ans*) and *I*, *i+1*|≠ Q, that contradicts to (i).

(TCQ_1 *S* TCQ_2). If Ans(Q, *i*) does not contain *ans*, according to Algorithm 1 it means that *ans* ∉ Ans(TCQ_2, *i*) × (Ans(TCQ_1, *i*) ⊳⊲ Ans(Q, *i-1*)). This, in turn, leads to the existence of a model *I* of a knowledge base K such that either *I*, *i* |≠TCQ_2(*ans*) or *I*, *i* |≠TCQ_1(*ans*) and *I*, *i-1*|≠ Q, that contradicts to (i).

The proof for the temporal operator O acting in the direction of future can be completed in the same manner.

CONCLUSIONS

Obtaining benefits from keeping a large evolving ABox of a knowledge base in a database and applying TBox of that knowledge base to obtain missing assertional axioms is one of the ways of dealing with complex evolving domains. It is interesting, due to high computational complexity of temporal conjunctive query answering in general, to find a balance between the expressivity of a query language and its practical applicability.

The main results of the paper are:

– for the point-based linear finite time structure elaborated is the language of unions of temporal conjunctive queries, which allows to evaluate atemporal unions of conjunctive queries at different time points. The language could serve as a simple yet expressive means to query a sequence of ABoxes sharing one TBox;

– proposed is an algorithm of answering a union of temporal conjunctive queries for the proposed language, which harnesses set-theoretic operations on atomic queries answer sets. Proved are its termination, soundness and completeness.

REFERENCES

1. Poggi A. Linking Data to Ontologies / A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, R. Rosati // J. on Data Semantics. — 2008. — X. — P. 133—173.

2. Baader F. A Novel Architecture For Situation Awareness Systems / F. Baader, A. Bauer, P. Baumgartner, A. Cregan, A. Gabaldon, K. Ji, K. Lee, D. Rajaratnam, R. Schwitter // Giese M., Waaler A. (eds.). — Proc. 18th International Conference on Automated Reasoning with Analytic Tableaux and RelatedMethods (Tableaux 2009). — LNCS. — 2009. — Vol. 5607. — P. 77—92.

3. The Description Logic Handbook: Theory, Implementation, and Applications / [Baader F., Calvanese D., McGuinness D. L. et al.]. — Cambridge University Press, 2003. — 555 p.

4. Tessaris S. Questions and Answers: Reasoning and Querying in Description Logic / Sergio Tessaris. —The University of Manchester, 2001. — 199 p.

5. Ortiz de la Fuente M. M. Query Answering in Expressive Description Logics Techniques and Complexity Results. PhD Thesis / Maria Magdalena Ortiz de la Fuente. — Technischen Universität Wien, Fakultät für Informatik, 2010. — 175 p.

6. Glimm B. Querying Description Logic Knowledge Bases. PhD Thesis / Birte Glimm. — The University of Manchester, 2007. — 192 p.

7. Motik B. Reasoning in Description Logics using Resolution and Deductive Databases / Boris Motik. — Univesität Karlsruhe, 2006. — 249 p.

8. On the Complexity of Temporal Query Answering. Technical report LTCS-Report 13-01 [electronic resource] / F. Baader, S. Borgwardt, M. Lippmann — Available at : http://lat.inf.tudresden.de/research/reports/2013/BaBoLi-LTCS-13-01.pdf.

9. Lipeck U. W. Transformation of Dynamic Integrity Constraints into Transaction Specifications / U. W. Lipeck // Theor. Comput. Sci. — 1990. — 76(1). — P. 115—143.

10. Schwiderski S. Monitoring Temporal Preconditions in a Behaviour Oriented Object Model / S. Schwiderski, T. Hartmann, G. Saake // Data Knowl. Eng. — 1994. — 14(2). — P. 143—186.

11. Lipeck U. W. Construction of Deterministic Transition Graphs from Dynamic Integrity Constraints / U.W. Lipeck, D. Feng. — LNCS. — 1989. — Vol. 344. — P.166—179.

12. Baier C. Principles of Model Checking / C. Baier, J.-P. Katoen. — The MIT Press, Cambridge, Massachusetts, 2008. — 975 p.

13. Abiteboul S. Foundations of Databases / S. Abiteboul, R. Hull, V. Vianu. — Addison-Wesley, 1995. — 685 p.

УДК 519.85

ПОЛУОПРЕДЕЛЕННОЕ ПРОГРАММИРОВАНИЕ ДЛЯ РЕШЕНИЯ ЗАДАЧ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ

Косолап А. И., д. ф.-м. н., профессор, Перетятько А. С., аспирант

Украинский государственный химико-технологический университет

В работе задачи комбинаторной оптимизации преобразуются к квадратичным. Затем используется полуопределенная релаксация для преобразования квадратичных задач к линейным полуопределенным задачам, в которых искомой является полуопределенная матрица. Для решения задачи полуопределенного программирования используется новый полуопределенный симплекс-метод. Найденное решение уточняется с помощью прямо-двойственного метода внутренней точки. Проведенные численные эксперименты подтверждают эффективность рассмотренных методов для решения комбинаторных задач оптимизации.

Ключевые слова: комбинаторная оптимизация, квадратичная оптимизация, полуопределенная релаксация, полуопределенное программирование, полуопределенный симплекс-метод, метод внутренней точки.

Косолап А. І., Перетятько А. С. НАПІВВИЗНАЧЕНЕ ПРОГРАМУВАННЯ ДЛЯ РОЗВЯЗКУ ЗАДАЧ КОМБІНАТОРНОЇ ОПТИМІЗАЦІЇ / Український державний хіміко-технологічний університет, Україна

У роботі задачі комбінаторної оптимізації перетворюються до квадратичних. Потім використовується напіввизначена релаксація для перетворення квадратичних задач до лінійних напіввизначених задач, у яких знаходиться напіввизначена матриця. Для розв'язання задач напіввизначеного програмування використовується новий напіввизначений симплекс-метод. Знайдений розв'язок уточнюється за допомогою прямо-двоїстого методу внутрішньої точки. Проведені чисельні експерименти підтверджують ефективність розглянутих методів для розв'язання комбінаторних задач оптимізації.

Ключові слова: комбінаторна оптимізація, квадратична оптимізація, напіввизначена релаксація, напіввизначене програмування, напіввизначений симплекс-метод, метод внутрішньої точки.

Kosolap A. I., Peretiatko A. S. SEMIDEFINITE PROGRAMMING FOR THE SOLUTION THE PROBLEMS OF COMBINATORIAL OPTIMIZATION / Ukrainian State University of Chemical Technology, Ukraine

The study deals with quadratic optimization problems with Boolean variables. Such problems arise in economics, management, technology, resource allocation, design, information systems, etc. The difficulty is that these problems may content lots of local minima and feasible set of these problems can be nonconvex and discrete. Most of these problems belong to NP-hard problems, and there are only exponential algorithms for their solving. One of the common approaches for solving this class of problems is semidefinite relaxation. In this case the quadratic function $x^T Ax$ is represented in the form Axx^T or $A \bullet X$, where A is symmetric matrix, and X is positive semidefinite matrix of rank one. This transformation allows to reduce the general quadratic problem to linear semidefinite optimization problem (SDP), in which the unknown variable is the