On Conditional Chase Termination

Gösta Grahne and Adrian Onet

Concordia University,Montreal, Canada, H3G 1M8
{grahne,a_onet}@cs.concordia.ca

Keywords: chase, closed world assumption, incomplete information, data exchange

1 Introduction

The chase procedure was initially developed for testing logical implication between sets of dependencies [3], for determining equivalence of database instances known to satisfy a given set of dependencies [14, 10], and for determining query equivalence under database constrains [2]. More recently the chase has been used to compute representative target solutions in data exchange. Intuitively, the data exchange problem consists of transforming a source database into a target database, according to a set of source to target dependencies describing the mapping between the source and the target. The set of dependencies may also include target dependencies, that is, constraints over for the target database. The source and the target schemas are considered to be distinct. Given a source instance I and a set Σ of dependencies, an instance J over the target schema is said to be a solution if $I \cup J$ satisfies all dependencies in Σ. One of the most important representations for this (usually infinite) set of solutions was introduced by Fagin et al. [6]. They considered the finite tableau obtained by chasing the initial instance with the dependencies. Such a tableau, if exists, was called a *universal solution*. In their paper, Fagin et al. showed that the universal solution is a good candidate for materializing the target. In particular, the universal solution can be used to compute certain answers to (unions of) conjunctive queries over the target instance. The universal solution is however not a good candidate for materialization in case we are dealing with nonmonotonic queries, or even conjunctive queries with inequalities. It was soon realized that a closed world semantics is needed. The first closed world approach in data exchange was introduced by Libkin [11] and extended by Hernich and Schweikardt in [9], and by Hernich in [8]. The paper by Deutsch et al. [5] also contained a proposal for dealing with nonmonotonic queries. In a previous paper [7], we analyzed these proposals and found that they can give unintuitive answers. To remedy this, we introduced the constructible solutions (constructible models) semantics, and showed that the space of constructible solutions can be represented by a conditional table, and obtained by a conditional chase, also given in [7].

Since the chase procedure was first adopted for data exchange one of the main problems was whether it terminates in a finite number of steps. Clearly, when considering only source to target dependencies the chase procedure terminates,

as there is no recursion involved. Unfortunately, adding even simple target constraints may lead to non-termination. For example, consider the data exchange setting given by one source to target dependency $\forall x, y : S(x, y) \rightarrow R(x, y)$, the target dependency $\forall x, y : R(x, y) \rightarrow \exists z\ R(y, z)$, and the source instance $I = \{S(a, b)\}$. It is easy to see that chasing I with the given dependencies is non-terminating, as at each step the target dependency will generate a new tuple, containing a fresh new labeled null, and that this new tuple will cause the target dependency to fire again.

This leads to a fundamental question: "When does the chase terminate?" This question was shown to be undecidable Deutsch et al. in [5]. Motivated by this negative result there has been several efforts to find sufficient conditions for the chase termination. The first of these was already given by Fagin et al. [6] who introduced the weakly acyclic sets of dependencies, and showed that the chase with such a set terminates. This was soon followed by several generalizations for which the chase is guaranteed to terminate [5, 13, 12, 15]. All of these previous results are for termination of the standard chase. The termination for variations of the chase did not receive much attention.

In this paper we show that our conditional chase is guaranteed to terminate for the richly acyclic dependencies introduced by Hernich and Schweikardt [9]. This class is slightly more restrictive than the weakly acyclic one. We also show that the termination of oblivious chase, introduced by Calì et al. [4], is strongly related to the termination of the standard chase, and that the oblivious chase always terminates with richly acyclic dependencies.

$\star$

The rest of this paper is organized as follows. In Section 2 we introduce basic notions used throughout the paper. In Section 3 we describe the constructible solutions semantics and the conditional chase procedure. Then, in Section 4 we show that the conditional chase with richly acyclic sets of dependencies is guaranteed to terminate. Conclusions are drawn in the last section.

2 Preliminaries

For required background knowledge the reader should consult [1]. Throughout the paper, the symbol ω will denote the set $\{0, 1, 2, \ldots\}$.

Relational instances. Let Cons be a countably infinite set of *constants*, usually denoted $a, b, c, \ldots$, possibly subscripted. From the domain Cons and a finite set $\mathbf{R}$ of relation names, we build up a Herbrand structure consisting of all expressions of the form $R(a_1, a_2, \ldots, a_k)$, where R is a "k-ary" relation name from $\mathbf{R}$, and the a_i's are values in Cons. Such expressions are called *ground atoms*, or *ground tuples*. A database instance I is then simply a finite set of ground tuples, e.g. $\{R_1(a_1, a_3), R_2(a_2, a_3, a_4)\}$. We denote the set of constants occurring in an instance I with $dom(I)$.

Relational tableaux. Let Vars be a countably infinite set, disjoint from the set of constants. Elements in Vars are called *variables*, and are denoted $X, Y, Z, \ldots$, possibly subscripted. We can then also allow *non-ground atoms*, i.e. expressions of the form $R(a, X, b, \ldots, X, Y)$. A *tableau* T is a finite set of atoms (ground, or non-ground). A non-ground atom represents a sentence where the variables are existentially quantified. The set of variables and constants in a tableau is denoted $dom(T)$.

Let T and U be tableaux. A mapping h from $dom(T)$ to $dom(U)$, that is the identity on Cons, extended component-wise, is called a *homomorphism* from T to U if $h(T) \subseteq U$. If there is an injective homomorphism h such that $h(T) = U$, and the inverse of h is a homomorphism from U to T, we say that T and U are *isomorphic*. An *endomorphism* on a tableau T is a mapping on $dom(T)$ such that $h(T) \subseteq T$. Finally, a *valuation* is a homomorphism whose image is contained in Cons.

Tuple generating dependencies. A *tuple generating dependency (tgd)* is a first order formula of the form

$$\forall \mathbf{x} : \alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \beta(\mathbf{x}, \mathbf{z})$$

where α is a conjunction of atoms, β is a single atom[1], and $\mathbf{x}$,$\mathbf{y}$ and $\mathbf{z}$ are sequences of variables. We assume that the variables occurring in dependencies are disjoint from all variables occurring in any tableaux under consideration. To emphasize this, we use lowercase letters for variables in dependencies. When α is the antecedent of a tgd, we shall sometimes conveniently regard it as a tableau, and refer to it as the *body* of the tgd. Similarly we refer to β as the *head* of the tgd. If there are no existentially quantified variables the dependency is said to be *full*, otherwise it is *embedded*. Frequently, we omit the universal quantifiers in tgd formulae. Also, when the variables are not relevant in the context, we denote a tgd $\alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\, \beta(\mathbf{x}, \mathbf{z})$ simply as $\alpha \rightarrow \beta$.

Let $\xi = \alpha \rightarrow \beta$ be a tuple generating dependency, and I an instance. Then we say that I *satisfies* ξ, if $I \models \xi$ in the standard model theoretic sense, or equivalently, if for every homomorphism h, such that $h(\alpha) \subseteq I$, there is an extension h' of h, such that $h'(\beta) \in I$.

The standard chase. The standard chase works on tableaux. Let Σ be a set of tgd's and T a tableau. A *trigger* for Σ on T is a pair (ξ, h), where $\xi = \alpha \rightarrow \beta \in \Sigma$, and h is a homomorphism such that $h(\alpha) \subseteq T$. The set of all triggers for Σ on T is denoted $trigg_\Sigma(T)$. It is easy to see that $trigg_\Sigma(T)$ is finite.

A trigger $(\xi, h) \in trigg_\Sigma(T)$, with $\xi = \alpha \rightarrow \beta$, is said to be *active* if there is no extension h' of for which $h'(\beta) \in T$. The *standard chase step* [6, 5] on a tableau T with such an active trigger results in a tableau $U = T \cup \{h'(\beta)\}$, where h' is an extension of h that assigns new fresh (uppercase) variables to the existential variables in β. The standard chase step is denoted $T \xrightarrow{\xi, h'} U$. We also say that

[1] Note that there is no loss of generality in assuming that the head consists of a single atom, cf. [4]

the trigger (ξ, h) was *fired* on T. The *standard chase* [6, 5] on a tableau T is defined as a sequence $T_0, T_1, T_2, \ldots, T_n, \ldots$ where $T = T_0$, and for each i we have that $T_i \xrightarrow{\xi,h'} T_{i+1}$, for some $(\xi, h) \in trigg_\Sigma(T_i)$. If there are no active triggers in $trigg_\Sigma(T_i)$ we let $T_{i+1} = T_i$. We say that the sequence *terminates* (in the finite) if $T_i = T_{i+1}$, for some $i \in \omega$. In the rest of this paper we consider all sequences to be *fair*, meaning that all active triggers either become inactivated or are eventually fired. The standard chase result for the sequence is $\bigcup_{i\in\omega} T_i$. Note that a chase sequence is not necessarily unique, since there is nondeterminism involved when choosing a trigger at each step. It is however known [6, 5] that all standard chase results are homomorphically equivalent. Let therefore $Chase_\Sigma^{\texttt{stand}}(T)$ denote one representative of this homomorphic equivalence class; in particular it denotes a finite representative, if one exists. Note that Deutsch et al. [5] have show that it is undecidable both whether some or whether all chase sequences terminate in the finite. There are however several classes of dependencies that guarantee termination of the standard chase, the most notable of them being the weakly acyclic dependencies for which all standard chase sequences terminate on all input instances. We shall return to this class of dependencies in Section 4.

3 The conditional chase

Deutsch et al. [5], building on the work on data exchange by Fagin et al. [6], regard the chase as a tool for computing a *universal model* for a tableau T and a set Σ of dependencies. A universal model for a theory is a model that has a homomorphism into every model of the theory. Such universal models have the desirable property that they are a sufficiently strong representative for computing certain answers to monotone queries on a database instance incompletely specified by $T \cup \Sigma$. A ground tuple t is in the certain answer to a query if t is in the answer to the query on *every* model of $T \cup \Sigma$. Recently it has been noted that for first order queries some sort of closed world assumption is required, as the set of models represented by a universal one is closed under homomorphic images, and thus amounts to an open world assumption. Several closed world assumptions have recently been proposed [11, 9, 5, 8]. These assumptions are scrutinized in [7], where a new closed world assumption, called *constructible models*, is put forward. This new semantics is based on the theory of incomplete information [10], where a tableau T is seen as a representative of a (possibly infinite) set of ground instances, or possible worlds, denoted $Rep(T)$. Formally, $Rep(T) = \{v(T) : v \text{ is a valuation}\}$. Then a tableau T is said to satisfy a set Σ of dependencies, if $I \models \Sigma$, for all $I \in Rep(T)$. A tableau $U = Chase_\Sigma^{\texttt{stand}}(T)$ is indeed a universal model for $T \cup \Sigma$, but there can be ground instances in $Rep(U)$ that do not satisfy Σ. For instance, let $T = \{R(a, b), R(X, c)\}$, and $\Sigma = \{R(x, y), R(y, z) \to S(x, z)\}$. Then $Chase_\Sigma^{\texttt{stand}}(T) = T$, $I = \{R(a, b), R(b, c)\} \in Rep(T)$, but $I \not\models \Sigma$. Consider now the nonmonotonic Boolean query $R(a, b) \wedge \neg S(a, c)$. Evaluated on T the certain answer will be **true**, which clearly should not be the case, as the query evaluates to **false** on the instance $I \cup \{S(a, c)\}$, and $I \cup \{S(a, c)\}$ is a model of $T \cup \Sigma$.

In [7] we show that the conditional tables [10] are sufficiently strong to exactly represent the aforementioned constructible models of $T \cup \Sigma$, and that such a conditional table can be obtained by a novel *conditional chase*. Next we briefly describe the constructible models, the conditional tables, and the conditional chase. More details can be found in [7].

Constructible models. Let I be a ground instance and Σ a set of tgd's. Consider a chase step on I with a tgd $\xi = \alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\ \beta(\mathbf{x}, \mathbf{z})$, and trigger (ξ, h). Instead of adding the tuple $\beta(h(\mathbf{x}), \mathbf{Z}')$, where $\mathbf{Z}'$ is a sequence of new fresh variables, we create a branch for each sequence $\mathbf{c}$ of constants and add the ground tuple $\beta(h(\mathbf{x}), \mathbf{c})$ to I. Continuing in this fashion we will have a *chase tree*, instead of a chase sequence. Notably, all nodes in the tree will be ground instances. The leaves of the chase tree form the set of constructible models of I and Σ, denoted $\Sigma(I)$. If the initial input to the chase is a tableau T, the constructible models of $T \cup \Sigma$ will be $\Sigma(\mathit{Rep}(T)) = \bigcup_{I \in \mathit{Rep}(T)} \Sigma(I)$.

Conditional tables. A *conditional table (c-table)* [10] is a pair (T, φ), where T is a tableau, and φ is a mapping that associates a *local condition* $\varphi(t)$ with each tuple $t \in T$. A (local) condition is a Boolean formula built up from atoms of the form $x = y$, $x = a$, and $a = b$ for $x, y \in \mathsf{Vars}$, and $a, b \in \mathsf{Cons}$. An atomic equality of the form $a = a$, for $a \in \mathsf{Cons}$ represents the logical constant **true**, and for two distinct constants a and b, the equality $a = b$ represents **false**. If $\varphi(t) = \mathbf{true}$, for all $t \in T$, the conditional table (T, φ) is denoted $(T, \mathbf{true})$, or sometimes simply with T. A conditional table (T, φ) *represents* a set of possible worlds (ground instances, complete databases). For this, let v be a valuation. Then

$$v(T, \varphi) = \{v(t) : t \in T, \text{ and } v\big(\varphi(t)\big) = \mathbf{true}\}.$$

The set of possible worlds represented by (T, φ) is

$$\mathit{Rep}(T, \varphi) = \{v(T, \varphi) : v \text{ is a valuation}\}.$$

Unifiers. Let T and U be tableaux. A *unifier* for T and U, if it exists, is a pair (θ_1, θ_2), where θ_1 is a homomorphism from $\mathit{dom}(T)$ to $\mathit{dom}(U)$, and θ_2 is an endomorphism on $\mathit{dom}(U)$, such that $\theta_1(T) = \theta_2(U)$. Note the asymmetrical role of T and U: a unifier for (T, U) is not necessarily a unifier for (U, T). The notion of a *most general unifier (mgu)* is defined in the obvious way, see [7]. For example, let $T = \{R(X, Y), R(Y, Z)\}$ and $U = \{R(a, V), R(b, W)\}$. Then (θ_1, θ_2), with $\theta_1 = \{X/a, Y/b,\ Z/W\}$ and $\theta_2 = \{V/b\}$ is an mgu for T and U. Another mgu for T and U is (γ_1, γ_2), with $\gamma_1 = \{X/b, Y/a,\ Z/V\}$ and $\gamma_2 = \{W/a\}$. We denote by $\mathit{mgu}(T, U)$ the set of all mgu's of T and U.

Triggers. Recall that for the classical chase a trigger for a tableau T is a pair (ξ, h), where $\xi = \alpha \rightarrow \beta$, and $h(\alpha) \subseteq T$. Let us now return to example given in the beginning of this section. We had $T = \{R(a, b), R(X, c)\}$, and $\Sigma = \{\xi\}$, where $\xi = \{R(x, y), R(y, z) \rightarrow S(x, z)\}$. We want to fire ξ on T resulting in conditional table $(\{R(a, b), R(X, c), S(a, c)\}, \varphi)$, where $\varphi(S(a, c))$ is $(X = b)$, and $\varphi(t)$ remains **true** for the two other tuples t. We can fire ξ on T provided that we unify X

with b. Formally, this is achieved by the trigger $(\xi, (\{x/a, y/b\}, \{X/b\}, T)$, where $(\{x/a, y/b\}, \{X/b\})$, is an mgu for the body of ξ, that is $\{R(x,y), R(y,z)\}$, and T. In general, the part of T that will be unified with the body of ξ is some subset of T. Let Σ be a set of tgd's, and (T, φ) a conditional table. A *trigger for* Σ *on* (T, φ) is a tuple $\tau = (\xi, \theta, T')$, where $\xi \in \Sigma$, and $\theta = (\theta_1, \theta_2)$ is an mgu that unifies the body of dependency ξ with T', where $T' \subseteq T$. The set $trigg_\Sigma(T, \varphi)$ contains the set of all triggers for Σ on (T, φ).

Example 1. Consider the conditional table (T, φ) with the following tabular representation:

t	$\varphi(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**

Let $\Sigma = \{\xi_1, \xi_2\}$, where tgd ξ_1 is $R(x,y), R(y,z) \rightarrow S(x,z)$ and tgd ξ_2 is $S(x,x) \rightarrow R(x,x)$. Then $trigg_\Sigma(T, \varphi) = \{\tau_1, \tau_2, \tau_3, , \tau_4, \tau_5\}$, with

$$\tau_1 = (\xi_1, (\{x/X, y/b, z/c\}, \{\}), \{R(X,b), R(b,c)\}),$$
$$\tau_2 = (\xi_1, (\{x/X, y/b, z/d\}, \{Y/b\}), \{R(X,b), R(Y,d)\}),$$
$$\tau_3 = (\xi_1, (\{x/b, y/c, z/d\}, \{Y/c\}), \{R(b,c), R(Y,d)\}),$$
$$\tau_4 = (\xi_1, (\{x/Y, y/d, z/b\}, \{X/d\}), \{R(Y,d), R(X,b)\}),$$
$$\tau_5 = (\xi_1, (\{x/c, y/c, z/b\}, \{X/c\}), \{R(b,c), R(X,b)\}).$$

Note that ξ_2, does not generate any triggers, as there are no tuples over relation name S. ◂

As applying a trigger on a c-table will generate new variables (unless the tgd is total), we need a mechanism for controlling this generation in such a way that the new variables are true representatives of the implicit Skolemization taking place. Let $\tau = (\xi, (\theta_1, \theta_2), T')$ be a trigger, where $\xi = \alpha(\mathbf{x}, \mathbf{y}) \rightarrow \exists \mathbf{z}\ \beta(\mathbf{x}, \mathbf{z})$. For each z in $\mathbf{z}$, consider a function $z \mapsto Z_\tau$, such that if $\rho = (\xi, (\theta_1, \theta_3), T'')$, then $Z_\tau = Z_\rho$. Clearly such a function exists (assuming `Vars` can be well ordered).

The following abbreviation will be useful. Let (T, φ) be a conditional table. Then we define

$$\curlywedge(T, \varphi) \ =_{\mathsf{def}} \ \bigwedge_{t \in T} \varphi(t).$$

Also, when θ is finite partial mapping from Vars to the set $\mathsf{Vars} \cup \mathsf{Const}$ we shall use the abbreviation

$$\curlywedge(\theta) \ =_{\mathsf{def}} \ \bigwedge_i X_i = \theta(X_i),$$

where the X_i's are all the variables in the domain of θ. We are now ready to define the conditional chase step.

Let $\tau = (\alpha \rightarrow \beta, (\theta_1, \theta_2), T') \in trigg_\Sigma(T, \varphi)$. We denote with θ_1' the extension of θ_1 that maps each existentially quantified variable z from the tgd to the variable Z_τ. We say that the conditional table (U, ϕ) *is obtained from* (T, φ) *by applying the trigger* τ if (U, ϕ) contains all the c-tuples from T together with their possibly modified local conditions, and possibly a new c-tuple, as follows:

If T contains a tuple t that is syntactically equal to $\theta_1'(\beta)$, ***then*** for (U, ϕ) the local condition of t is changed to

$$\phi(t) =_{\mathsf{def}} \varphi(t) \vee \text{ⱥ}(\tau),$$

where $\text{ⱥ}(\tau)$ denotes [2] the formula $\text{ⱥ}(T', \varphi) \wedge \text{ⱥ}(\theta_2)$,
else add the c-tuple $\theta_1'(\beta)$ with local condition

$$\phi(\theta_1'(\beta)) =_{\mathsf{def}} \text{ⱥ}(\tau).$$

If (U, ϕ) is obtained from (T, φ) in this way we write $(T, \varphi) \to_\tau (U, \phi)$, and call the transformation a *conditional chase micro-step.*

Having a table (T, φ) and a finite set Σ of tgd's, the set of triggers $trigg_\Sigma(T, \varphi)$ is obviously finite. Consider a sequence

$$(T, \varphi) \to_{\tau_1} (U_1, \varphi_1) \to_{\tau_2} \cdots \to_{\tau_n} (U_n, \varphi_n),$$

where $\tau_1, \tau_2, \ldots, \tau_n$ is an ordering of all the triggers from the set $trigg_\Sigma(T, \varphi)$. We call it a *conditional chase micro-sequence for* (T, φ) *using* $trigg_\Sigma(T, \varphi)$.

Example 2. Let us continue with the c-table from Example 1. The c-table (U_1, ψ_1) in Figure 1 is obtained from (T, φ) in one chase micro-step by applying the trigger τ_1. That is $(T, \varphi) \to_{\tau_1} (U_1, \psi_1)$. Next, by applying trigger τ_2 to (U_1, ψ_1), we obtain (U_2, ψ_2). Note that the last tuple has a local condition generated by the substitution $\{Y/b\}$ from τ_2. In the same way we apply τ_3 to (U_2, ψ_2), obtaining (U_3, ψ_3). Then by applying τ_4, c-table (U_4, ψ_4) is obtained, to which we apply τ_5, obtaining c-table (U_5, ψ_5). This gives the following sequence of chase micro-steps:

$$(T, \varphi) \to_{\tau_1} (U_1, \psi_1) \to_{\tau_2} (U_2, \psi_2) \to_{\tau_3} (U_3, \psi_3) \to_{\tau_4} (U_4, \psi_4) \to_{\tau_5} (U_5, \psi_5),$$

where the c-tables in question are displayed in Figure 1.◂

We note that the order in which the triggers are applied in a conditional micro-sequence does affect the outcome, but we shall see that in the end the order will not matter.

After a micro-sequence, it is not guaranteed that the result will satisfy the dependencies. Then additional triggers will be generated. We therefore abstract a micro-sequence into a macro-step, as follows.

Let (T, φ) and (U, ϕ) be conditional tables, and Σ be a set of tgd's. We write $(T, \varphi) \Rightarrow_\Sigma (U, \phi)$, if (U, ϕ) is obtained from (T, φ) by applying all micro-steps generated from $trigg_\Sigma(T, \varphi)$. The transformation from (T, φ) to (U, ϕ) is called a *conditional chase macro-step using* Σ.

[2] Note that the formula $\text{ⱥ}(\tau)$ is uniquely determined by τ. Intuitively it represents the ("abducted") conditions induced by τ and necessary for the body α to apply in the standard sense.

Fig. 1. Conditional tables from Example 2

(U_1, ψ_1)

t	$\psi_1(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**
$S(X,c)$	**true**

(U_2, ψ_2)

t	$\psi_2(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**
$S(X,c)$	**true**
$S(X,d)$	$Y = b$

(U_3, ψ_3)

t	$\psi_3(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**
$S(X,c)$	**true**
$S(X,d)$	$Y = b$
$S(b,d)$	$Y = c$

(U_4, ψ_4)

t	$\psi_4(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**
$S(X,c)$	**true**
$S(X,d)$	$Y = b$
$S(b,d)$	$Y = c$
$S(Y,b)$	$X = d$

(U_5, ψ_5)

t	$\psi_5(t)$
$R(X,b)$	**true**
$R(b,c)$	**true**
$R(Y,d)$	**true**
$S(X,c)$	**true**
$S(X,d)$	$Y = b$
$S(b,d)$	$Y = c$
$S(Y,b)$	$X = d$
$S(b,b)$	$X = c$

The macro-step is monotone, that is, if $(T, \varphi) \Rightarrow_\Sigma (U, \phi)$, then $T \subseteq U$ and φ logically implies ϕ, which we by a slight abuse of notation denote $\varphi \subseteq \phi$. We next introduce the concept of a conditional chase-sequence.

Let (T, φ) be a c-table and Σ a set of tgd's. A sequence

$$(T_0, \varphi_0), (T_1, \varphi_1), \ldots, (T_n, \varphi_n), \ldots$$

of c-tables, inductively constructed as

$$\begin{aligned} (T_0, \varphi_0) &= (T, \varphi), \\ (T_{i+1}, \varphi_{i+1}) &= (U, \phi), \text{ where } (T_i, \varphi_i) \Rightarrow_\Sigma (U, \phi), \\ (T_\omega, \varphi_\omega) &= \bigcup_{i \in \omega} (T_i, \varphi_i), \end{aligned}$$

is called a *conditional chase sequence* associated with (T, φ) and Σ. If there is an $i \in \omega$, such that $\mathit{Rep}(T_i, \varphi_i) = \mathit{Rep}(T_{i+1}, \varphi_{i+1})$, we say that the conditional chase sequence *terminates* (in the finite).

Example 3. Continuing example 2, let $(T_0, \varphi_0) = (T, \varphi)$ and $(T_1, \varphi_1) = (U_5, \psi_5)$. Then $(T_0, \varphi_0) \Rightarrow_\Sigma (T_1, \varphi_1)$. Next, $\mathit{trigg}_\Sigma(T_1, \varphi_1) = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6, \tau_7, \tau_8\}$, where $\tau_1, \ldots, \tau_5$ are as before, and

$$\begin{aligned} \tau_6 &= (\xi_2, (\{x/d, y/d\}, \{X/d\}), \{S(X,d)\}), \\ \tau_7 &= (\xi_2, (\{x/b, y/b\}, \{Y/b\}), \{S(Y,b)\}), \\ \tau_8 &= (\xi_2, (\{x/b, y/b\}, \{\}), \{S(b,b)\}), \end{aligned}$$

By applying the triggers in the order $\tau_1, \tau_2, \ldots, \tau_8$ we obtain another micro-sequence ending in (T_2, φ_2), shown in Figure 2. In other words, $(T_1, \varphi_1) \Rightarrow_\Sigma (T_2, \varphi_2)$. Note that the conditional chase process is not finished yet, as for table (T_2, φ_2), beside the previous triggers the following new triggers need to be considered as well:

$$\begin{aligned} \tau_9 &= (\xi_1, (\{x/X, y/b, z/b\}, \{\}), \{R(X,b), R(b,b)\}), \\ \tau_{10} &= (\xi_1, (\{x/b, y/b, z/b\}, \{X/b\}), \{R(b,b), R(X,b)\}), \\ \tau_{11} &= (\xi_1, (\{x/Y, y/d, z/d\}, \{\}), \{R(Y,d), R(d,d)\}), \\ \tau_{12} &= (\xi_1, (\{x/d, y/d, z/d\}, \{Y/d\}), \{R(d,d), R(Y,d)\}). \end{aligned}$$

Fig. 2. Conditional tables (T_2, φ_2) and (T_3, φ_3) from Example 3

(T_2, φ_2)

t	$\varphi_2(t)$
$R(X, b)$	**true**
$R(b, c)$	**true**
$R(Y, d)$	**true**
$S(X, c)$	**true**
$S(X, d)$	$Y = b \vee Y = b$
$S(b, d)$	$Y = c \vee Y = c$
$S(Y, b)$	$X = d \vee X = d$
$R(d, d)$	$(Y = b \vee Y = b) \wedge (X = d)$
$R(b, b)$	$((X = d \vee X = d) \wedge (Y = b)) \vee (X = c)$

(T_3, φ_3)

t	$\varphi_3(t)$
$R(X, b)$	**true**
$R(b, c)$	**true**
$R(Y, d)$	**true**
$S(X, c)$	**true**
$S(X, d)$	$Y = b$
$S(b, d)$	$Y = c$
$S(Y, b)$	$X = d$
$R(d, d)$	$Y = b \wedge X = d$
$R(b, b)$	$(X = d \wedge Y = b) \vee (X = c)$
$S(X, b)$	$(X = d \wedge Y = b) \vee (X = c)$
$S(Y, d)$	$Y = b \wedge X = d$

By applying $\tau_1, \ldots, \tau_{12}$ the micro-sequence will end up in the c-table (T_3, φ_3), in Figure 2. Thus $(T_0, \varphi_0), (T_1, \varphi_1), (T_2, \varphi_2), (T_3, \varphi_3)$ is (the finite part of) a conditional chase sequence associated with (T, φ) and Σ. For readability, (T_3, φ_3) is shown in a "cleaner" isomorphically equivalent form.◂

The following result shows that no matter what order we choose to apply the triggers in the micro-sequences, the results of the corresponding conditional chase sequences are equivalent.

Theorem 1. *Let (T, φ) be a conditional table and Σ a set of tgd's. Then, let $(T_0, \varphi_0), (T_1, \varphi_1), \ldots, (T_n, \varphi_n), \ldots$ and $(T'_0, \varphi'_0), (T'_1, \varphi'_1), \ldots, (T'_n, \varphi'_n), \ldots$ be two distinct conditional chase sequences with different orders of triggers in their micro-sequences. Then $Rep(T_\omega, \varphi_\omega) = Rep(T'_\omega, \varphi'_\omega)$. Even more, $T_\omega = T'_\omega$.*

The theorem means that distinct orders in the micro-sequences only affect the syntactic form of the local conditions. The atoms in the table will be syntactically equal, and the corresponding local conditions equivalent.

We shall use the notation $Chase^{\mathtt{cond}}_{\Sigma}(T, \varphi)$ for one representative of $(T_\omega, \varphi_\omega)$ in the conditional chase sequence, starting with (T, φ) and Σ. We conclude this section by stating the main property of the conditional chase. The theorem means that the conditional chase computes exactly the set of constructible models.

Theorem 2. $Rep(Chase^{\mathtt{cond}}_{\Sigma}(T, \varphi)) = \Sigma(Rep(T, \varphi))$.

4 When does the conditional chase terminate?

We will show that the conditional chase with richly acyclic set of tgd's always terminates. The class of richly acyclic [9] sets of tgd's is slightly smaller than the well known class of weakly acyclic sets of tgd's. To prove our result we need the oblivious chase, introduced by Calì et al. [4].

The oblivious chase. The oblivious chase proceeds as the standard chase, except that each dependency is fired once for each trigger, irrespectively of whether the trigger is active or not. As in the standard chase, we assume that all the dependencies are applied fairly in the oblivious chase process. Fairness of the oblivious chase means that each trigger, active or not, is eventually applied. The details of how to construct a fair oblivious chase sequence are described in [4]. In case we use a fair oblivious chase, the result is unique up to isomorphism. We denote with $Chase_{\Sigma}^{\mathtt{obl}}(T)$ one representative instance of the result. Clearly there are cases where the standard chase terminates, but the oblivious does not. However, Calì et al. [4] show that the results of the standard and the oblivious chases are homomorphically equivalent, even if one (or both) of them is nonterminating.

Enrichment of dependencies. In order to relate termination of the standard and the oblivious chase, we introduce a transformation, called *enrichment*, that takes a tuple generating dependency $\xi = \alpha(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \beta(\mathbf{x},\mathbf{z})$ and converts it into the tgd $\hat{\xi} = \alpha(\mathbf{x},\mathbf{y}) \rightarrow \exists \mathbf{z}\ \beta(\mathbf{x},\mathbf{z}), H(\mathbf{x},\mathbf{y})$, where H is a new relational symbol that does not appear in $\mathbf{R}$. For a set Σ of tgd's, the transformed set is $\widehat{\Sigma} = \{\hat{\xi} : \xi \in \Sigma\}$. We now have

Theorem 3. *$Chase_{\Sigma}^{\mathtt{obl}}(T)$ terminates if and only if $Chase_{\widehat{\Sigma}}^{\mathtt{stand}}(T)$ terminates.*

Proof. Let Σ be a set of tgd's, and suppose that there is a tableau T for which the standard chase with $\widehat{\Sigma}$ does not terminate. This means that there is an infinite standard chase sequence

$$T = T_0 \xrightarrow{\hat{\xi}_0, h'_0} T_1 \xrightarrow{\hat{\xi}_1, h'_1} \ldots\ldots \xrightarrow{\hat{\xi}_{n-1}, h'_{n-1}} T_n \xrightarrow{\hat{\xi}_n, h'_n} \ldots.$$

Let α_i denote the antecedent of $\hat{\xi}_i$ Then $h_i(\alpha_i) \subseteq T_i$, for all $i \in \omega$. Since α_i also is the antecedent of ξ_i, we have that

$$T = U_0 \xrightarrow{\xi_0, h'_0} U_1 \xrightarrow{\xi_1, h'_1} \ldots\ldots \xrightarrow{\xi_{n-1}, h'_{n-1}} U_n \xrightarrow{\xi_n, h'_n} \ldots,$$

where U_i is T_i with all atoms over H deleted, is an infinite oblivious chase sequence with Σ on T.

Suppose then that there is a tableau T for which the oblivious chase with Σ does not terminate. Then there is an infinite oblivious chase sequence

$$T = T_0 \xrightarrow{\xi_0, h'_0} T_1 \xrightarrow{\xi_1, h'_1} \ldots\ldots \xrightarrow{\xi_{n-1}, h'_{n-1}} T_n \xrightarrow{\xi_n, h'_n} \ldots.$$

Let $U_0 = T_0$, and for all $i \in \omega$, let

$$U_{i+1} = T_{i+1} \cup \{\beta(h'_i(\mathbf{x}), h'_i(\mathbf{z})), H(h'_i(\mathbf{x}), h'_i(\mathbf{y}))\}.$$

We claim that

$$T = U_0 \xrightarrow{\hat{\xi}_0, h'_0} U_1 \xrightarrow{\hat{\xi}_1, h'_1} \ldots\ldots \xrightarrow{\hat{\xi}_{n-1}, h'_{n-1}} U_n \xrightarrow{\hat{\xi}_n, h'_n} \ldots$$

is an infinite standard chase sequence with $\widehat{\Sigma}$ on T. Suppose it is not. Then there must be an $i \in \omega$, such that the standard chase step cannot be applied with h_i and $\hat{\xi}_i$ on U_i. Let $\xi_i = \alpha(\mathbf{x},\mathbf{y}) \rightarrow \beta(\mathbf{x},\mathbf{z})$. Then $\hat{\xi}_i = \alpha(\mathbf{x},\mathbf{y}) \rightarrow \beta(\mathbf{x},\mathbf{z}), H(\mathbf{x},\mathbf{y})$. If $\hat{\xi}_i$ cannot be applied with h_i on U_i, there exists an extension h_i^e of h_i such that $\beta(h_i^e(\mathbf{x}), h_i^e(\mathbf{z})))$ and $H(h_i^e(\mathbf{x}), h_i^e(\mathbf{y})))$ are in U_i. Since h_i^e is an extension of h_i, it follows that $h_i^e(\mathbf{x}) = h_i(\mathbf{x})$ and $h_i^e(\mathbf{y}) = h_i(\mathbf{y})$, meaning that $H(h_i(\mathbf{x}), h_i(\mathbf{y}))) \in U_i$. Because atoms over H are only introduced by the chase, it means that the homomorphism h_i has already been applied with $\hat{\xi}_i$ earlier in the standard chase sequence. But then h_i must also have been applied with ξ_i at the same earlier stage in the oblivious chase sequence. This is a contradiction, since it entails that ξ_i would have been fired twice with h_i in the oblivious chase sequence.∎

Weakly and richly acyclic sets of tgd's. For a set Σ of tgd's over a database schema $\mathbf{R}$ the *dependency graph* of Σ is the directed graph that has as vertices (R,i), where $R \in \mathbf{R}$, and $i \in \{1,\ldots,arity(R)\}$. The graph has two types of edges:

1. *Regular edges.* There is a regular edge between vertices (R,i) and (S,j) if there a tgd in Σ that has a variable y that appears both in position (R,i) in the body, and in position (S,j) in the head.
2. *Existential edges.* There is an existential edge between vertices (R,i) and (S,j) if there is a tgd in Σ that has a variable y that appears in position (R,i) of the body and in some position in the head, and an existentially quantified variable z that appears in position (S,j) in the head.

A set of Σ of tgd's is said to be *weakly acyclic* if the dependency graph of Σ does not have any cycles containing an existential edge [6]. In particular, Fagin et al. have shown

Lemma 1. [6] *Let Σ be a weakly acyclic set of tgd's. Then $Chase_\Sigma^{\mathtt{stand}}(T)$ terminates for all tableaux T.*

Hernich and Schweikardt [9] defined a slight restriction on the weakly acyclic sets of dependencies as follows: If the previous dependency graph is extended by also adding an existential edge from position (R,i) and (S,j) whenever there is a dependency in $\xi \in \Sigma$ that has a variable x that appears in position (R,i) of the body and an existentially quantified variable z appears in position (S,j) in the head of ξ. The newly created graph is called *extended dependency graph.* A set of Σ of tgd's is said to be *richly acyclic* if the extended dependency graph of Σ does not have any cycles containing an existential edge. The following lemma is immediate.

Lemma 2. *Let Σ be a richly acyclic set of tgd's. Then $\widehat{\Sigma}$ is weakly acyclic.*

From Theorem 3, and Lemmas 1 and 2, we can conclude

Corollary 1. *Let Σ be a richly acyclic set of tgd's. Then $Chase_\Sigma^{\mathtt{obl}}(T)$ terminates on all tableaux T.*

It is easy to see that whenever the conditional chase terminates, the oblivious chase also terminates. The reverse is however not true.

Proposition 1. *There exists a set Σ of tgd's and a tableau T, such that the oblivious chase with Σ terminates on all instances (including T), but the conditional chase with Σ does not terminate on $(T, \mathbf{true})$.*

Proof: Let $\Sigma = \{R(x,y,y) \rightarrow \exists z,v\ R(x,z,v)\}$. Using the technique in [12] it can easily be shown that the standard chase with $\widehat{\Sigma}$ terminates on all instances. From this, and Theorem 3 it follows that the oblivious chase with Σ terminates on all instances. Consider then the conditional chase starting with c-table $\{R(a,b,b)\}$ (the local condition of $R(a,b,b)$ is **true**). The result of this chase can be represented by the infinite conditional table below.■

t	$\varphi(t)$
$R(a,b,b)$	**true**
$R(a,X_1,X_2)$	**true**
$R(a,X_3,X_4)$	$X_1 = X_2$
$R(a,X_5,X_6)$	$(X_1 = X_2) \wedge (X_3 = X_4)$
$\ldots$	$\ldots$

In order to obtain the sufficient condition for the conditional chase termination we introduce our second rewriting of a set of tgd's Σ, called *disjoining*, and denoted $\ddot{\Sigma}$. Given a tgd ξ, we denote with $\ddot{\xi}$ the same dependency where for each variable x, that is repeated in the body, we replace all occurrences, except the first, of x in the body with a fresh new universally quantified variable. For instance, if $\xi = R(x,y,x), R(y,z,x) \rightarrow \exists v,w\ R(x,v,w)$, then $\ddot{\xi} = R(x,y,x_1), R(y_1,z,x_2) \rightarrow \exists v,w\ R(x,v,w)$, where the subscripted variables are fresh. The set $\ddot{\Sigma}$ is then $\{\ddot{\xi} : \xi \in \Sigma\}$.

From the observation that a difference between the conditional and the oblivious chase is that the former may fire based on unifiers (θ_1, θ_2), and the latter based only on homomorphisms θ_1, we obtain the following lemma.

Lemma 3. *$Chase_{\Sigma}^{\mathtt{cond}}(T,\varphi)$ terminates if $Chase_{\ddot{\Sigma}}^{\mathtt{obl}}(T)$ terminates.*

From the observation that the extended dependency graph for $\ddot{\Sigma}$ can be obtained from the extended dependency graph of Σ by deleting edges we get

Lemma 4. *If Σ is a richly acyclic set of tgd's, then $\ddot{\Sigma}$ is also richly acyclic.*

We now have the main result of this section

Theorem 4. *Let Σ be a set of richly acyclic set of tgd's and (T,φ) a c-table. Then $Chase_{\Sigma}^{\mathtt{cond}}(T,\varphi)$ terminates.*

Proof. By Lemma 4, the set $\ddot{\Sigma}$ is also richly acyclic, and by Lemma 2, the set $\widehat{\ddot{\Sigma}}$ is weakly acyclic. By [6], $Chase_{\widehat{\ddot{\Sigma}}}^{\mathtt{stand}}(T, \mathbf{true})$ terminates. By Corollary 1, $Chase_{\ddot{\Sigma}}^{\mathtt{obl}}(T, \mathbf{true})$ also terminates, and then by Theorem 3, $Chase_{\Sigma}^{\mathtt{cond}}(T,\varphi)$ terminates.■

5 Conclusions

In this paper we have shown that the conditional chase with a richly acyclic set of dependencies terminates on all instances. This class is based on the well known weakly acyclic class of dependencies that ensures the standard chase termination on all instances. One may extend the class of dependencies for which the conditional chase terminates, by considering classes of dependencies for which the standard chase terminates (such as super weakly acyclic, safe constraints, C-stratified) and then using our characterization of the oblivious chase termination (Theorem 3) and by suitable extensions of Lemma 3.

Due to the space restrictions complexity results are not included. In the full version of this paper we show that for richly acyclic sets of dependencies the problem of checking if a tuple is the certain answer to a given first order query is coNP-complete and remains polynomial for unions of conjunctive queries.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Aho, A.V., Beeri, C., Ullman, J.D.: The theory of joins in relational databases. ACM Trans. Database Syst. 4(3), 297–314 (1979)
3. Beeri, C., Vardi, M.Y.: A proof procedure for data dependencies. J. ACM 31(4), 718–741 (1984)
4. Calì, A., Gottlob, G., Kifer, M.: Taming the infinite chase: Query answering under expressive relational constraints. In: KR. pp. 70–80 (2008)
5. Deutsch, A., Nash, A., Remmel, J.B.: The chase revisited. In: PODS. pp. 149–158 (2008)
6. Fagin, R., Kolaitis, P.G., Miller, R.J., Popa, L.: Data exchange: Semantics and query answering. In: ICDT. pp. 207–224 (2003)
7. Grahne, G., Onet, A.: Closed world chasing. In: LID. pp. 7–14 (2011)
8. Hernich, A.: Answering non-monotonic queries in relational data exchange. In: ICDT. pp. 143–154 (2010)
9. Hernich, A., Schweikardt, N.: Cwa-solutions for data exchange settings with target dependencies. In: PODS. pp. 113–122 (2007)
10. Imielinski, T., Jr., W.L.: Incomplete information in relational databases. J. ACM 31(4), 761–791 (1984)
11. Libkin, L.: Data exchange and incomplete information. In: PODS. pp. 60–69 (2006)
12. Marnette, B.: Generalized schema-mappings: from termination to tractability. In: PODS. pp. 13–22 (2009)
13. Meier, M., Schmidt, M., Lausen, G.: On chase termination beyond stratification. PVLDB 2(1), 970–981 (2009)
14. Mendelzon, A.O.: Database states and their tableaux. In: XP2 Workshop on Relational Database Theory (1981)
15. Spezzano, F., Greco, S.: Chase termination: A constraints rewriting approach. PVLDB 3(1), 93–104 (2010)