On the Translatability of View Updates

Enrico Franconi and Paolo Guagliardo

KRDB Research Centre, Free University of Bozen-Bolzano, Italy
<surname>@inf.unibz.it

Abstract We revisit the view update problem and the abstract functional framework by Bancilhon and Spyratos in a setting where views and updates are exactly given by functions that are expressible in first-order logic. We give a characterisation of views and their inverses based on the notion of definability, and we introduce a general method for checking whether a view update can be uniquely translated as an update of the underlying database under the constant complement principle. We study the setting consisting of a single database relation and two views defined by projections and compare our general criterion for translatability with the known results for the case in which the constraints on the database are given by functional dependencies. We extend the setting to any number of projective views, full dependencies (that is, egd's and full tgd's) as database constraints, and classes of updates rather than single updates.

1 Introduction

Updating a database by means of a set of views is a challenging task that requires updates performed on the views to be “translated” into suitable updates of the underlying database, in order to consistently propagate the changes on the views to the base relations over which the views are defined.

A general and precise understanding of the view update problem is due to the seminal work [2] by Bancilhon and Spyratos (B&S), who devise a functional framework in which they formalise the problem and provide an elegant solution to it. They introduce the notion of *view complement*, representing what is missing from a view to have the same informative content of the underlying database. Moreover, they introduce the *constant complement* principle, establishing that the changes made on a view must not influence the content of its complement. B&S do not provide actual methods for checking the translatability of updates and computing their translations, asserting that “computational algorithms (if they exist) must be sought in specific problems”.

In the context of SQL databases, Lechtenbörger [12] gives a characterisation of the constant complement principle in terms of “undo” operations, showing that view updates are translatable under constant complement precisely if users have the chance to undo all effects of their updates by using further view updates. It is then argued that testing whether this holds could be an alternative to checking whether users can observe all effects of their updates in the view schema.

Gottlob et al. [9] extend the results of [2] to the class of so-called *consistent views*, which properly contains the views translating under constant complement.

The main difference is that, during the translation of an update on a consistent view, the complement is not required to remain invariant, but it is allowed to "decrease" according to a suitable partial order. Indeed, when the partial order is the equality, the framework coincides with the one in [2].

Cosmadakis and Papadimitriou [6] consider a restricted setting that consists of a single database relation and two views defined by projections. They provide necessary and sufficient conditions for the translatability of insertions, deletions and replacements under constant complement w.r.t. a specific database instance and when the constraints on the database are given by functional dependencies (fd's). To the best of our knowledge, [6] is the only comprehensive work in which the framework by B&S is applied to a relational setting. We discuss in detail this application scenario in Sec. 4, where we extend the setting to any number of views defined by projections, rather than just two, and more expressive database constraints, namely full dependencies (egd's and full tgd's), and to classes of updates rather than single updates.

In [2], the view update problem is formalised at a high level of abstraction, where views and updates are arbitrary functions, of which no constructive characterisation is given, as indeed one might not even be possible. In this paper, we consider the view update problem at a lower level of abstraction, by revisiting B&S' framework in a setting where views and updates are exactly given by functions that are expressible in first-order logic (FOL). Under certain conditions, this class of functions can be constructively characterised through the notion of *logical definability*, in terms of which we introduce a general method for checking the translatability of arbitrary FO-expressible view updates. With this work we mainly contribute the following:

- a general framework for view updating that is based on the notion of determinacy and constructively revisits [2] in a setting with constraints;
- a general method, applicable whenever the inverse of a view mapping can be constructively characterised, for checking whether a view update can be propagated to the underlying database in a unique way;
- a practical application setting consisting of projective views of a single database relation, that, although still very limited, extends the known existing results [6] to more expressive database constraints, any number of acyclic projective views and more general view updates.

The paper is organised as follows: in Sec. 2 we introduce some notation and basic definitions; in Sec. 3 we present our logic-based framework and characterise when and whether a FO-expressible view update is uniquely translatable under constant complement; in Sec. 4 we study the case considered in [6] and generalise the results to a more general setting; we conclude in Sec. 5 by pointing out some future research directions. Proofs and more detailed examples are given in the full version [7].

2 Preliminaries

An *n-ary relation* on a set A, where $n \in \mathbb{N}_1$ denotes the *arity* of the relation, is a subset of the Cartesian product A^n, that is, a set of n-tuples of elements of A.

A *signature* is a finite set $\mathcal{S}$ of relation symbols and each symbols $S \in \mathcal{S}$ has an associated arity denoted by $\mathsf{arity}(S)$. A *relational structure* s over a signature $\mathcal{S}$ is a pair $\langle \Delta^s, \cdot^s \rangle$ where Δ^s is a (possibly infinite) domain of objects and $\cdot^s$ is an *interpretation function* that associates each symbol $S \in \mathcal{S}$ with a relation S^s on Δ^s, called the *extension* of S, of appropriate arity. For two relational structures $s = \langle \Delta, \cdot^s \rangle$ and $t = \langle \Delta, \cdot^t \rangle$ over two disjoint signatures $\mathcal{S}$ and $\mathcal{S}'$, respectively, $s \uplus t = \langle \Delta, \cdot^s \cup \cdot^t \rangle$ is a relational structure over $\mathcal{S} \cup \mathcal{S}'$. A *constraint* is a closed formula φ in (some fragment of) FOL. The set of relation symbols occurring in φ is denoted by $\mathsf{sig}(\varphi)$ and φ is said to be *over a signature* $\mathcal{S}$ if $\mathsf{sig}(\varphi) \subseteq \mathcal{S}$. We extend $\mathsf{sig}(\cdot)$ to sets of constraints in the natural way. Sequences (i.e., tuples) are denoted with an overline, e.g., $\overline{x}$, and $\overline{x}[k]$ denotes the k-th element in $\overline{x}$. The number of elements in $\overline{x}$ is denoted by $|\overline{x}|$ and, when $\overline{x}$ is a sequence of variables, $\mathsf{var}(\overline{x})$ denotes the set of variables occurring in it.

A *renaming* over a signature $\mathcal{S}$ is a bijective function $\mathsf{ren} : \mathcal{S} \to \mathcal{S}'$, where $\mathcal{S}'$ is a signature disjoint with $\mathcal{S}$. We extend $\mathsf{ren}(\cdot)$ to signatures, relational structures and (sets of) constraints in the natural way. For instance, given a constraint φ, $\mathsf{ren}(\varphi)$ is obtained from φ by replacing every occurrence of each relation symbol $r \in \mathsf{sig}(\varphi)$ with $\mathsf{ren}(r)$. Clearly, for a set Σ of constraints over $\mathcal{S}$ and a relational structure s over $\mathsf{ren}(\mathcal{S})$, we have that $s \models \mathsf{ren}(\Sigma)$ iff $\mathsf{ren}^{-1}(s) \models \Sigma$.

A *database schema* is a signature $\mathcal{R}$ of *database symbols* and a *database state* is a relational structure over $\mathcal{R}$. A *view schema* is a signature $\mathcal{V}$ of *view symbols* not occurring in $\mathcal{R}$ and a *view state* is a relational structure over $\mathcal{V}$. We denote the set of all database (resp., view) states by $\mathbf{S}$ (resp., $\mathbf{T}$). For a database state $s \in \mathbf{S}$ and a view state $t \in \mathbf{T}$ having the same domain, the relational structure $s \uplus t$ is called a *global state* over $\mathcal{R} \cup \mathcal{V}$. We consider a satisfiable finite set Σ of *global constraints* over the signature $\mathcal{R} \cup \mathcal{V}$. A database state s (resp., view state t) is *Σ-consistent* iff there exists a view state t (resp., database state s) with the same domain such that the global state $s \uplus t$ is a model of Σ. We denote the set of Σ-consistent database states (resp., view states) by $\mathbf{S}_\Sigma$ (resp., $\mathbf{T}_\Sigma$). When Σ is understood from the context, we refer to Σ-consistent states generically as *globally consistent* states or states that are *consistent with the global constraints.*

Definition 1 (View under constraints). *A* view *from* $\mathcal{R}$ *to* $\mathcal{V}$ under con*straints Σ is a total mapping $f \colon \boldsymbol{S}_\Sigma \to \boldsymbol{T}_\Sigma$ s.t. $s \uplus f(s) \models \Sigma$ for every $s \in \boldsymbol{S}_\Sigma$.*

Since $\mathcal{R}$ and $\mathcal{V}$ are disjoint, every model of Σ has the form $s \uplus t$, where $s \in \mathbf{S}$ and $t \in \mathbf{T}$ are (globally consistent) states with the same domain. We say that $V \in \mathcal{V}$ is *implicitly defined* by the symbols in $\mathcal{R}$ under Σ if for every pair of global states $s \uplus t$ and $s \uplus t'$ satisfying Σ, it is the case that $V^t = V^{t'}$. In other words, every two models of Σ (with the same domain) agreeing on the interpretation of the symbols in $\mathcal{R}$ also agree on the interpretation of V.

Definition 2. $\mathcal{R}$ defines $\mathcal{V}$ *under Σ (written $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$) iff, for every $s \in \boldsymbol{S}$ and $t, t' \in \boldsymbol{T}$, it is the case that $t = t'$ whenever $s \uplus t \models \Sigma$ and $s \uplus t' \models \Sigma$.*

In particular, $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ means that every $V \in \mathcal{V}$ is implicitly defined by $\mathcal{R}$ under Σ. We use $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$, $\mathcal{R} \not\twoheadrightarrow_\Sigma \mathcal{V}$ and $\mathcal{V} \not\twoheadrightarrow_\Sigma \mathcal{R}$ with the obvious meaning.

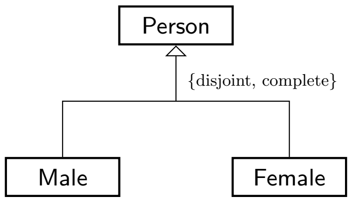

Figure 1: A UML class diagram where each class is implicitly defined by the others (e.g., knowing all the persons and who the males are, we also implicitly know who the females are).

A fundamental result by Beth [4] states that in FOL, whenever V is implicitly defined by $\mathcal{R}$ under constraints Σ, there always exists an *explicit definition* of V in terms of $\mathcal{R}$, that is, a formula φ equivalent to V under Σ with $\mathsf{sig}(\varphi) \subseteq \mathcal{R}$.

Example 1. Consider the UML class diagram depicted in Figure 1, stating that (1) "a Male is a Person", (2) "a Female is a Person", (3) "Male is disjoint with Female" (4) "a Person is Male or Female". This can be expressed as a first-order logic theory Σ over $\{\mathsf{Male}, \mathsf{Female}, \mathsf{Person}\}$ consisting of the following formulae:

$$\forall x \,.\ \mathsf{Male}(x) \rightarrow \mathsf{Person}(x)\ \ ; \tag{1}$$

$$\forall x \,.\ \mathsf{Female}(x) \rightarrow \mathsf{Person}(x)\ \ ; \tag{2}$$

$$\forall x \,.\ \mathsf{Male}(x) \rightarrow \neg\,\mathsf{Female}(x)\ \ ; \tag{3}$$

$$\forall x \,.\ \mathsf{Person}(x) \rightarrow \mathsf{Male}(x) \vee \mathsf{Female}(x)\ \ . \tag{4}$$

Under the constraints in Σ, whenever we "fix" who the persons and the males are, we also *implicitly* determine who the females are. Indeed, when $\mathsf{Person}(x)$ and $\mathsf{Male}(x)$ are considered database predicates, $\mathsf{Female}(x)$ is *explicitly* defined as the view $\mathsf{Person}(x) \wedge \neg\,\mathsf{Male}(x)$. That is, under the given constraints, a female is exactly a person who is not male.

In general, there might exist more than one view mapping satisfying a given set of constraints. An important connection between definability and views under constraints is that the mapping is unique exactly when each view symbol can be defined in terms of the database symbols under the given constraints.

Theorem 1. *$\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$ iff there is one and only one view from $\mathcal{R}$ to $\mathcal{V}$ under Σ.*

The above theorem gives a characterisation of the views that are expressible by means of constraints in FOL. In what follows, we write $\mathcal{R} \twoheadrightarrow^{f}_{\Sigma} \mathcal{V}$ to indicate that $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$ and f is the (one and only) view *induced by the constraints Σ* from $\mathcal{R}$ to $\mathcal{V}$ in light of Theorem 1. The *surjection induced by* (or *surjective restriction of*) a function f is the surjective function obtained from f by restricting its codomain to its image. We use concatenation to indicate composition, e.g., fg denotes the composition of f with g.

The framework we will present in the next section is based on the notion of logical definability, and relies on the fact that whenever something is implicitly defined it is possible to find its explicit definition as a view in FOL [8]. Unfortunately, Beth's result [4] holds when instances are *unrestricted* (i.e., allowed to be possibly infinite), while it fails [10] when considering finite instances only, which is the usual case of interest in database applications. Clearly, when something holds for unrestricted models it also holds in particular for finite models, therefore our framework is sound. On the other hand, if something holds on finite models, it does not necessarily hold on all models, hence our approach might in general be incomplete, in the sense that we may fail to discover invertible view mappings and translatable view updates that are such only on finite models, because our techniques require these properties to hold on unrestricted models. We say that a problem is *finitely controllable* iff it holds true for unrestricted models whenever it holds true for finite ones (the vice versa is always true). We will prove that the implicit definability problem in the application scenario we discuss in detail in Sec. 4 is finitely controllable; so, in this case, our results are going to be sound and complete.

3 A Logic-Based Framework for View Updates

In this section, we present our framework for view updating based on the notion of definability in logic. We first revisit some of the formal definitions given in [2], adapting them to our logic-based setting. Next, we show that a view induced by a set of constraints is invertible exactly when each database symbol is implicitly defined by the view symbols under the same constraints. We then give a general criterion for translatability in terms of a special set of view constraints that are satisfied exactly by each and every FO-expressible translatable view update. We conclude by discussing the notion of view complement and show how the results on translatability extend to the case of translation under constant complement.

A *database update* (resp., *view update*) is a function $d\colon \mathbf{S} \to \mathbf{S}$ (resp., $u\colon \mathbf{T} \to \mathbf{T}$) associating each database state (resp., view state) with another one having the same domain. The sets of all the database and view updates are denoted by $U_{\mathcal{R}}$ and $U_{\mathcal{V}}$, respectively.

Given a view under constraints and a view update, we want to find a suitable database update that propagates the changes to the base relations in a consistent way. More precisely, the view update should be translated as a database update that brings the database to a new state from which, through the view mapping, we reach exactly the updated view state. In addition, unjustified and unnecessary changes in the database are to be avoided, in the sense that if the view update does not modify the view state, then the database update must not modify the corresponding database state either.

Definition 3 (Translation). *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $d \in U_{\mathcal{R}}$ and $u \in U_{\mathcal{V}}$. The database update d is a* translation *of u (w.r.t. f) if and only if 1) $uf = fd$ and 2) $\forall s \in \boldsymbol{S}_{\Sigma}$, $uf(s) = f(s) \implies d(s) = s$.*

A translation can only exist if the updated view state lies in the image of f; otherwise, there would be no chance of reaching the new view state by means of

f from some database state. A view update that can be translated into a suitable database update, i.e., for which there exists a translation, is called *translatable.*

Definition 4 (Translatability). *Let $f\colon \boldsymbol{S}_\Sigma \to \boldsymbol{T}_\Sigma$ be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ. A view update $u \in U_\mathcal{V}$ is* translatable *(w.r.t. f) if and only if for each $s \in \boldsymbol{S}_\Sigma$ there exists $s' \in \boldsymbol{S}_\Sigma$ such that $f(s') = uf(s)$.*

Translatability of view updates ensures that there exists a translation, but does not rule out the possibility that there might be more than one. To avoid ambiguity, we are only interested in view updates that are *uniquely translatable*, that is, for which there exists one and only one translation. For injective views, a view update is translatable if and only if it is uniquely translatable, and the following theorem [2] gives a characterisation of the unique database update into which a translatable view update can be translated when the view is injective.

Theorem 2. *Let f be an injective view under Σ, let $u \in U_\mathcal{V}$ be translatable and let $d \in U_\mathcal{R}$. Let $\hat{f}$ denote the surjection induced by f and let $\hat{u}$ be obtained from u by restricting its domain and codomain to $f(\boldsymbol{S}_\Sigma)$.*[1] *Then, d is a translation of u if and only if $d = \hat{f}^{-1}\hat{u}\hat{f}$.*

It is possible to invert a view induced by a set of constraints iff the database symbols are implicitly defined by the view symbols under the same constraints, in which case the inverse is also effectively computable. In such a situation, the constraints induce two mappings, one from the database states to the view states and one in the opposite direction, which are indeed one the inverse of the other.

Lemma 1. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$. Then, f is surjective; and f is injective iff $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$.*

Theorem 3. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$. Then, f is invertible iff $\mathcal{V} \twoheadrightarrow^h_\Sigma \mathcal{R}$, and $h = f^{-1}$.*

Given Σ over $\mathcal{R} \cup \mathcal{V}$, we denote by $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ the problem of checking whether $R \in \mathcal{R}$ is implicitly defined by the symbols in $\mathcal{V}$ under Σ, which amounts to checking whether the following entailment holds:

$$\Sigma \cup \mathsf{ren}(\Sigma) \models \forall \overline{x} \left(R(\overline{x}) \leftrightarrow R'(\overline{x}) \right) \ ,$$

where ren is a renaming over $\mathcal{R}$ and $R' = \mathsf{ren}(R)$. Note that, as R' is simply a renaming of R, for symmetric reasons it is sufficient to check the entailment of only one of the implications on the r.h.s. of (3). Given a view f from $\mathcal{R}$ to $\mathcal{V}$ induced by Σ, we denote by $\mathsf{invert}(\mathcal{V}, \Sigma)$ the problem of determining whether f is invertible, which amounts to checking whether $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ for each $R \in \mathcal{R}$ (where $\mathcal{R}$ is given by $\mathsf{sig}(\Sigma) \setminus \mathcal{V}$).

In the following, we provide an interesting characterisation of when a view update is translatable. The idea consists essentially in imposing additional constraints on the view schema so that every *legal* view update is translatable and vice versa. Whenever $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$ there exists an explicit definition for each of the

[1] As u is assumed to be translatable, $u\big(f(\mathbf{S}_\Sigma)\big) \subseteq f(\mathbf{S}_\Sigma)$.

database symbols in terms of the view symbols. By substituting every occurrence in Σ of each $R \in \mathcal{R}$ with its explicit definition we obtain a set $\widetilde{\Sigma}_{\mathcal{V}}$ of constraints over $\mathcal{V}$ which we call $\widetilde{\Sigma}_{\mathcal{V}}$ the $\mathcal{V}$-*embedding* of Σ.

Theorem 4. *Let $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and let $t \in \boldsymbol{T}$. Then, t is Σ-consistent iff $t \models \widetilde{\Sigma}_{\mathcal{V}}$.*

The $\mathcal{V}$-embedding of a set Σ of global constraints is a set of view constraints having the same "restrictiveness" of the whole Σ, but with the advantage that they can be checked locally on the view schema. This is of particular relevance for surjective views, in which case it turns out that such constraints ensure the translatability of every view update satisfying them and enforce every translatable view update to be legal with respect to them.

Theorem 5. *Let f be a surjective view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and let $u \in U_{\mathcal{V}}$. Then, u is translatable if and only if $u(t) \models \widetilde{\Sigma}_{\mathcal{V}}$ for every $t \in \boldsymbol{T}_{\Sigma}$.*

In other words, when updating a view state that is legal w.r.t. $\widetilde{\Sigma}_{\mathcal{V}}$, we need to make sure that we always end up in another legal view state.

Let ren be a renaming over $\mathcal{R} \cup \mathcal{V}$ and let Ξ be a set of constraints over $\mathcal{V} \cup \mathcal{V}'$ s.t. $\mathcal{V} \twoheadrightarrow_{\Xi} \mathcal{V}'$, where $\mathcal{V}' = \mathsf{ren}(\mathcal{V})$. Let $\widetilde{\Xi}_{\mathcal{V}}$ be obtained from Ξ by replacing, for each symbol $V' \in \mathcal{V}'$, every occurrence of the predicate $V'(\overline{x})$ with its explicit definition in terms of $\mathcal{V}$. Then, the function ξ induced by Ξ *expresses* a view update $u \colon \mathbf{T} \to \mathbf{T}$ iff $\widetilde{\Xi}_{\mathcal{V}}$ is valid.[2] Indeed, in such a case, ξ takes a view state t over $\mathcal{V}$ and returns an updated view state $\xi(t)$ over $\mathcal{V}'$. The view update u expressed by Ξ is then the function associating each $t \in \mathbf{T}$ with $\mathsf{ren}^{-1}\big(\xi(t)\big)$. Note that every set of constraints Ξ over $\mathcal{V} \cup \mathcal{V}'$ that expresses a view update is equivalent to a set of constraints over $\mathcal{V} \cup \mathcal{V}'$ consisting of exactly one formula for each $V \in \mathcal{V}$ of the form $\forall \overline{x}\,\big(V(\overline{x}) \leftrightarrow \phi_V(\overline{x})\big)$, with $\mathsf{sig}\big(\phi_V(\overline{x})\big) \subseteq \mathcal{V}'$.

For a view symbol V, insertion and deletion of a tuple $\overline{x}$ are represented by the following open formulae:

$$\begin{aligned}\mathsf{insert}_V(\overline{x}) &\equiv \forall \overline{y}\,\big[V'(\overline{y}) \leftrightarrow \big(V(y) \vee \overline{y} = \overline{x}\,\big)\big] \;; \\ \mathsf{delete}_V(\overline{x}) &\equiv \forall \overline{y}\,\big[\big(V'(\overline{y}) \to V(\overline{y})\big) \wedge \big(V(\overline{y}) \to \big[V'(\overline{y}) \vee \overline{y} = \overline{x}\,\big]\big) \wedge \neg V'(\overline{x})\big] \;.\end{aligned}$$

Any update that does not modify V can be represented by the closed formula $\mathsf{noop}_V \equiv \forall \overline{x}\,\big(V'(\overline{x}) \leftrightarrow V(\overline{x})\big)$, while the replacement of a tuple $\overline{x}$ with a tuple $\overline{y}$ is expressed by the open formula $\mathsf{replace}_V(\overline{x}, \overline{y})$ defined by:

$$\big[\big(\neg V(\overline{x}) \vee \overline{x} = \overline{y}\,\big) \to \mathsf{noop}_V\big] \wedge \big[\big(V(\overline{x}) \wedge \overline{x} \neq \overline{y}\,\big) \to \mathsf{delete}_V(\overline{x}) \wedge \mathsf{insert}_V(\overline{y})\big] \;.$$

In our formalism we can also directly express transactional updates, in the sense of sequences of updates that are applied one after the other. For instance, suppose we want to insert a tuple $\overline{a}$ into the extension of V and then delete tuple $\overline{b}$ from it. Note that the update $\mathsf{insert}_V(\overline{a}) \wedge \mathsf{delete}_V(\overline{b})$ implies $V'(\overline{a})$ and $\neg V'(\overline{b})$ (where V' represents V after the update), hence it is inconsistent if $\overline{a} = \overline{b}$. The correct way to represent the two sequential updates as a transaction is to consider the

[2] This is needed to ensure that Ξ does not impose constraints on the view schema.

update $\mathsf{insert}_V(\overline{a}) \wedge \mathsf{delete}_{V'}(\overline{b})$, where $\mathsf{delete}_{V'}(\overline{b})$ is applied on V', which is the result of applying $\mathsf{insert}_V(\overline{a})$ on V.

From Theorem 5, we get the following characterisation of the translatability of those view updates that are expressible, as described, in FOL.

Theorem 6. *Let f be a surjective view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and let $u \in U_{\mathcal{V}}$ be expressed by Ξ. Then, u is translatable iff $\widetilde{\Sigma}_{\mathcal{V}} \cup \Xi \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big)$.*

Under the assumptions of the above theorem, the view f is injective by Lemma 1. Hence, by Theorem 2 every translatable view update u has the unique translation $f^{-1}uf$. However, we might not be able to actually compute f^{-1} unless $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$, in which case Theorem 3 ensures that the inverse of f is the view from $\mathcal{V}$ to $\mathcal{R}$ induced by Σ. When $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$, $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and Ξ expresses a translatable view update u, we have that $\mathcal{V} \twoheadrightarrow_{\Xi} \mathsf{ren}(\mathcal{V})$ and $\mathsf{ren}(\mathcal{V}) \twoheadrightarrow_{\mathsf{ren}(\Sigma)} \mathsf{ren}(\mathcal{R})$, therefore the unique translation of u is the database update expressed by the set Υ such that $\mathcal{R} \twoheadrightarrow_{\Upsilon} \mathsf{ren}(\mathcal{R})$, obtained by replacing in $\mathsf{ren}(\Sigma)$ every occurrence of $\mathsf{ren}(V)$ with its definition in terms of $\mathcal{V}$ and, in turn, every occurrence of V with its definition in terms of $\mathcal{R}$.

Given the $\mathcal{V}$-embedding $\widetilde{\Sigma}_{\mathcal{V}}$ of a set of constraints Σ over $\mathcal{R} \cup \mathcal{V}$ inducing an invertible view, and given a set of constraints Ξ expressing a view update $u \in U_{\mathcal{V}}$, $\mathsf{translat}(\Xi, \widetilde{\Sigma}_{\mathcal{V}})$ is the problem of determining whether u is translatable, that is, from Theorem 6, whether $\widetilde{\Sigma}_{\mathcal{V}} \cup \Xi \models \mathsf{ren}(\widetilde{\Sigma}_{\mathcal{V}})$. Note that a view update which is not translatable in general might still be translated when it is applied on a given view state. For example, the insertion of a specific tuple of values is unlikely to be translatable for all possible view states, but it might be such for a given (legal) view state. This related problem, indicated with $\mathsf{translat}(t, \Xi, \widetilde{\Sigma}_{\mathcal{V}})$, of checking whether the update is translatable w.r.t. a view state t satisfying $\widetilde{\Sigma}_{\mathcal{V}}$, that is, whether $u(t) \models \widetilde{\Sigma}_{\mathcal{V}}$. Clearly, $\mathsf{translat}(\Xi, \widetilde{\Sigma}_{\mathcal{V}})$ amounts to checking $\mathsf{translat}(t, \Xi, \widetilde{\Sigma}_{\mathcal{V}})$ for each $t \in \mathbf{T}$ such that $t \models \widetilde{\Sigma}_{\mathcal{V}}$.

We conclude the section by briefly discussing the notion of *complement* and the principle of *translation under constant complement* introduced in [2]. Lack of injectivity in a view f causes loss of information, due to the fact that distinct database states are mapped to the same view state. A *complement* of f is a view g operating on the same domain of f and capable of distinguishing between distinct database states which f maps to the same view state. If g is a complement of f, then f is a complement of g (that is, the notion of complement is symmetric), therefore we sometimes simply say that two views f and g are "complementary".

A view complement provides additional information that, together with the original view, allows to reconstruct the entire database. Two views f and g under constraints Σ and Γ, respectively, can be combined in a natural way into a single view under $\Sigma \cup \Gamma$, which we call the *union* of f and g, written as $f \uplus g$. It turns out that there is a close connection between complementarity and injectivity of views, given by the fact that two views under constraints and with the same domain are complementary if and only if their union is injective.

Thus, by means of a view complement we can recover information missing from a lossy view and gain injectivity, which gives us the possibility of using the results presented earlier. However, following the rationale that the only purpose for which a view complement is made available is that of allowing for a lossy view to be updatable, we demand that the information it provides be invariant during the update process. In other words, view updates must never modify, neither directly nor indirectly, any data that belongs to the view complement. For complementary views f and g, a view update on f is *g-translatable* (or *translatable under constant complement g*), if it is translatable (in the sense of Definition 4) and in addition does not modify the extension of the symbols belonging to the complement g. In general, there might be more than one complement of a given view, and an update is g-translatable or not depending on the particular complement g we consider. Therefore, the choice of a complement defines an "update policy", assigning unambiguous semantics to the view updates.

Given updates u and v on f and g, respectively, with some abuse of notation we denote by $u \uplus v$ the combined update on $f \uplus g$. Then, the following theorem establishes the relationship between translatability w.r.t. a view under constant complement and translatability w.r.t. the union of a view and its complement.

Theorem 7. *Let f and g be complementary, let $u \in U_{\mathcal{V}}$ and let v be the identity. Then, u is g-translatable w.r.t. f if and only if $u \uplus v$ is translatable w.r.t. $f \uplus g$.*

This essentially means that, in general, given an injective view f from $\mathcal{R}$ to $\mathcal{V}$ and a set $\mathcal{C} \subseteq \mathcal{V}$ of view symbols whose extension is required to remain constant, we need to check for the translatability of view updates $u \in U_{\mathcal{V}}$ such that $V^t = V^{u(t)}$ for every $V \in \mathcal{C}$ and every $t \in \mathbf{T}$.

4 Translatable Updates on Projective Views

In this section, we discuss in some detail a setting consisting of a single database relation and views defined by projections. It turns out that, when the database constraints are full dependencies, the view mapping defined by the projections is invertible iff the database relation can be reconstructed by their natural join. We point out that invertibility of views under constraints is finitely controllable in this case, which is a generalisation of the setting studied in [6] where only two views are considered, and that our general criterion for translatability of updates w.r.t. an instance under egd's and full tgd's subsumes the conditions given in [6] for the case in which the database constraints are fd's only.

The general setting we consider here consists of a single database relation over a universal set of attributes U and views defined by projections on subsets $X_1, \ldots, X_n$ of U. We assume w.l.o.g. that U is a totally ordered set and denote by $\mathsf{apos}(A)$ the position of attribute A within U. For a subset X of U, $\mathsf{apos}(X)$ denotes the set $\{\mathsf{apos}(A) \mid A \in X\}$. Let $\mathcal{R} = \{R\}$ and $\mathcal{V} = \{V_1, \ldots, V_n\}$, where R and each $V_i \in \mathcal{V}$ have arities $|U|$ and $|X_i|$, respectively. Each position p in R is associated with the (one and only) attribute $A \in U$ for which $\mathsf{apos}(A) = p$. Then, a projection on $X \subseteq U$ is expressed by the open formula $\mathsf{project}_X(\overline{x}) \equiv \exists \overline{y}\, R(\overline{w})$, where $\overline{x}$, $\overline{y}$ and $\overline{w}$ are sequences of distinct variables such that $\mathsf{var}(\overline{x}) \cap \mathsf{var}(\overline{y}) = \varnothing$, $\mathsf{var}(\overline{w}) = \mathsf{var}(\overline{x}) \cup \mathsf{var}(\overline{y})$ and $|\overline{w}| = |\overline{x}| + |\overline{y}|$. In addition, all variables from

$\mathsf{var}(\overline{x})$ are required to appear in $\overline{w}$ at positions $\mathsf{apos}(X)$ and in the same order in which they appear in $\overline{x}$.

The set of global constraints we consider is $\Sigma = \Sigma_{\mathcal{R}} \cup \Sigma_{\mathcal{RV}}$, where $\Sigma_{\mathcal{R}}$ is a set of constraints over $\mathcal{R}$ and $\Sigma_{\mathcal{RV}}$ consists of a formula of the form $\forall \overline{x} \left(V_i(\overline{x}) \leftrightarrow \mathsf{project}_{X_i}(\overline{x})\right)$ for each $V_i \in \mathcal{V}$. Let $f\colon \mathbf{S}_\Sigma \to \mathbf{T}_\Sigma$ be the view induced by Σ and observe that, since there are no constraints on the view schema, every database state that satisfies $\Sigma_{\mathcal{R}}$ is Σ-consistent, therefore S_Σ coincides with the set of legal database states. Moreover, being a view induced by constraints, f is surjective, and it is invertible iff $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$.

We denote by $\mathsf{pos}(V_i, p)$ the position of R corresponding to the p-th position of V_i and $\mathsf{pos}(V_i)$ denotes the set $\{\mathsf{pos}(V_i, p) \mid 1 \leq p \leq \mathsf{arity}(V_i)\}$, that is, the set of positions of R on which V_i projects. As an example, for $V_i(x_1, x_2)$ defined as $\exists y_1\, R(y_1, x_1, x_2)$, $\mathsf{pos}(V_i, 2) = 3$ because the variable x_2 in the second position of V_i occurs in R at position 3, and $\mathsf{pos}(V_i) = \{2, 3\}$. We say that $\mathcal{V}$ is *acyclic* if the hypergraph with $\{1, \dots, \mathsf{arity}(R)\}$ as set of nodes and $\{\mathsf{pos}(V_i) \mid V_i \in \mathcal{V}\}$ as set of hyperegedes contains no cycles (see Section 6.4 in [1]).

Let us first consider the case, studied in [6], in which there are only two view symbols, that is, $\mathcal{V} = \{V_1, V_2\}$. Rephrasing the definition given in [6], we say that the view symbols V_1 and V_2 are *complementary* if for every two legal *finite* database states s and s' for which ${V_1}^{f(s)} = {V_1}^{f(s')}$ and ${V_2}^{f(s)} = {V_2}^{f(s')}$ it is the case that $R^s = R^{s'}$. Note that the notion of complementarity is equivalent to the implicit definability of R from V_1 and V_2 under Σ when considering finite states only. In other words, V_1 and V_2 are complementary iff $\mathsf{impl\text{-}def}_{\mathrm{fin}}(R, \mathcal{V}, \Sigma)$, where $\mathsf{impl\text{-}def}_{\mathrm{fin}}$ is $\mathsf{impl\text{-}def}$ restricted to finite models. It is shown in [6] that, when $\Sigma_{\mathcal{R}}$ consists of functional and join dependencies, V_1 and V_2 are complementary if and only if $\Sigma_{\mathcal{R}}$ finitely implies the jd $\bowtie[X_1, X_2]$, that is, the extension of R can always be reconstructed from the extensions of V_1 and V_2 by means of natural join. Now, since unrestricted and finite implication of a jd from a set of fd's and jd's coincide [1], complementarity in the finite case implies complementarity in the unrestricted case, and the same goes for implicit definability. Therefore, when $\Sigma_{\mathcal{R}}$ consists only of fd's and jd's, $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ and $\mathsf{impl\text{-}def}_{\mathrm{fin}}(R, \mathcal{V}, \Sigma)$ coincide, that is, $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ is finitely controllable and, in turn, also $\mathsf{invert}(\mathcal{V}, \Sigma)$.

The above results can be extended to the more general setting where $\mathcal{V} = \{V_1, \dots, V_n\}$ with $n \geq 2$ and $\Sigma_{\mathcal{R}}$ consists of full dependencies, provided that $\mathcal{V}$ is acyclic. Indeed, in [3] it is shown that under full dependencies the decomposition of a database relation into a set of acyclic projections is lossless if and only if the reconstruction operator is the natural join. Losslessness (of vertical decompositions) and complementarity (of projective views) are equivalent notions, hence the result in [3] properly generalizes the one in [6], as fd's and jd's are a special case of egd's and full tgd's, respectively, and two projections are always acyclic. Since for full dependencies finite and unrestricted implication coincide [1], $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ is finitely controllable also in this extended setting. Moreover, we know that whenever R is implicitly defined by $\mathcal{V}$ under Σ, the extension of R can be reconstructed from the extensions of the n symbols in $\mathcal{V}$ by natural join, that is, Σ entails the equivalence $\forall \overline{x} \left(R(\overline{x}) \leftrightarrow V_1(\overline{x}_1) \wedge \dots \wedge V_n(\overline{x}_n)\right)$ where

$\overline{x}$ and $\overline{x}_1, \ldots, \overline{x}_n$ are sequences of variables such that $\mathsf{var}(\overline{x}) = \bigcup_{i=1}^{n} \mathsf{var}(\overline{x}_i)$, $\overline{x}_i[p] = \overline{x}_j[q]$ iff $\mathsf{pos}(V_i, p) = \mathsf{pos}(V_j, q)$, and $\overline{x}[p] = \overline{x}_i[q]$ iff $p = \mathsf{pos}(V_i, q)$. In other words, variables corresponding to the same position in R must coincide. Note that the above equivalence is well-defined iff $\{1, \ldots, \mathsf{arity}(R)\} = \bigcup_{i=1}^{n} \mathsf{pos}(V_i)$, that is, the projections cover the positions of R entirely.

We now turn to the problem of checking translatability under constant complement w.r.t. a view state in the extended setting with n complementary projective views. Thus, let $\mathcal{V} = \{V_1, \ldots, V_n\}$ with $n \geq 2$ and let $\mathcal{C} \subseteq \mathcal{V}$ be the set of symbols constituting the complement, that is, whose extension must remain invariant during the update. In general, here $\Sigma_{\mathcal{R}}$ is a set of full dependencies. In our approach, testing whether a view update u is translatable w.r.t. a finite legal view state t can be done in PTIME in the size of $u(t)$,[3] provided that $u(t)$ is also finite, by checking that $u(t)$ satisfies the $\mathcal{V}$-embedding $\widetilde{\Sigma}_{\mathcal{V}}$ of Σ, which is obtained by replacing every occurrence of $R(\overline{x})$ in Σ with its explicit definition, that is, the formula $V_1(\overline{x}_1) \land \cdots \land V_n(\overline{x}_n)$.

For the special case when $n = 2$, necessary and sufficient conditions for the translatability w.r.t. an instance of insertions, deletions and replacements in the extension of V_1 while keeping the extension of V_2 constant are given in [6], with the further restriction that $\Sigma_{\mathcal{R}}$ consists of fd's only. As an exercise, it is easy to check that the conditions given separately in [6] for the translatability w.r.t. an instance of insertions, deletions and replacements can be obtained by spelling out in each case our general criterion for translatability w.r.t. a view state, which subsumes all of them, modulo the fact that we allow for more general database constraints rather than just fd's and that we consider insertions (deletions) of possibly (non-)existing tuples and replacements of a tuple with possibly the same one.[4] We give an idea of how our criterion corresponds to the conditions of [6] in the case of insertions by means of an example.

Example 2. Let $U = \{E, D, P, S, M\}$, where E stands for *Employee*, D for *Department*, P for *Position*, S for *Salary* and M for *Manager*. Let $<$ be a total order on U s.t. $E < D < P < S < M$, thus $\mathsf{apos}(E) = 1$, $\mathsf{apos}(D) = 2$, $\mathsf{apos}(P) = 3$, $\mathsf{apos}(S) = 4$ and $\mathsf{apos}(M) = 5$. Let $\mathcal{V} = \{V_1, V_2\}$ and $\Sigma_{\mathcal{RV}}$ consist of:

$$\forall x_1, x_2, x_3 \quad \big(V_1(x_1, x_2, x_3) \quad \leftrightarrow \exists y_1, y_2\, R(x_1, x_2, x_3, y_1, y_2)\big) \;; \tag{5a}$$

$$\forall x_1, x_2, x_3, x_4\, \big(V_2(x_1, x_2, x_3, x_4) \leftrightarrow \exists y_1 \quad R(x_1, x_2, y_1, x_3, x_4)\big) \;; \tag{5b}$$

that is, the two view symbols are defined by projections on EDP and $EDSM$. Let $\Sigma_{\mathcal{R}}$ consists of the following constraints:

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \land R(x_1, x_2, x_3', x_4', x_5') \rightarrow x_3 = x_3'\big) \;; \tag{6a}$$

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \land R(x_1', x_2', x_3, x_4', x_5') \rightarrow x_4 = x_4'\big) \;; \tag{6b}$$

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \land R(x_1', x_2, x_3', x_4', x_5') \rightarrow x_5 = x_5'\big) \;; \tag{6c}$$

[3] This is the data complexity of testing whether a finite relational structure is a model of a FOL theory.

[4] For instance, condition (b) of Theorem 3 in [6] is necessary only due to the assumption that the tuple to be inserted is not already present in the view extension.

where $\overline{x}$ is the sequence of all the variables appearing in each case. Equations (6a), (6b) and (6c) express the fd's $ED \to P$, $P \to S$ and $D \to M$, respectively. It can be verified that $\Sigma = \Sigma_{\mathcal{R}} \cup \Sigma_{\mathcal{RV}}$ implies the jd $\bowtie[EDP, EDSM]$, that is:

$$\forall \overline{x} \left(R(x_1, x_2, x_3, x_4, x_5) \leftrightarrow V_1(x_1, x_2, x_3) \wedge V_2(x_1, x_2, x_4, x_5) \right) \ . \tag{7}$$

By substituting every occurrence of $R(x_1, x_2, x_3, x_4)$ in (5a), (5b), (6a), (6b) and (6c) with the explicit definition $V_1(x_1, x_2, x_3) \wedge V_2(x_1, x_2, x_4, x_5)$ we obtain:

$$\forall x_1, x_2, x_3 \quad \left(V_1(x_1, x_2, x_3) \quad \to \exists y_1, y_2 \, V_2(x_1, x_2, y_1, y_2) \right) \ ; \tag{8a}$$

$$\forall x_1, x_2, x_3, x_4 \left(V_2(x_1, x_2, x_3, x_4) \to \exists y_1 \quad V_1(x_1, x_2, y_1) \quad \right) \ ; \tag{8b}$$

$$\forall \overline{x} \left(V_1(x_1, x_2, x_3) \wedge V_1(x_1, x_2, x_3') \quad \to x_3 = x_3' \right) \ ; \tag{8c}$$

$$\begin{aligned} &\forall \overline{x} \left(V_1(x_1, x_2, x_3) \wedge V_2(x_1, x_2, x_4, x_5) \right. \\ &\quad \left. \wedge V_1(x_1', x_2', x_3) \wedge V_2(x_1', x_2', x_4', x_5') \to x_4 = x_4' \right) \ ; \end{aligned} \tag{8d}$$

$$\forall \overline{x} \left(V_2(x_1, x_2, x_3, x_4) \wedge V_2(x_1', x_2, x_3', x_4') \to x_4 = x_4' \right) \ ; \tag{8e}$$

together constituting the $\mathcal{V}$-embedding $\widetilde{\Sigma}_{\mathcal{V}}$ of Σ. Note that, while the fd's (6a) and (6c) on R are preserved as the fd's (8c) and (8e) on V_1 and V_2, respectively, the fd (6b) becomes a genuine egd, namely (8d), on V_1 and V_2.

Let $\overline{a} = \langle e, d, p, s \rangle$ and consider the view update u, expressed by $\{\mathsf{noop}_{V_1}, \mathsf{insert}_{V_2}(\overline{a})\}$, that inserts the tuple $\overline{a}$ into the extension of V_2 while the extension of V_1 remains unchanged. Given a view state t satisfying $\widetilde{\Sigma}_{\mathcal{V}}$, u is translatable w.r.t. t iff $u(t)$ satisfies $\widetilde{\Sigma}_{\mathcal{V}}$ too, where $u(t)$ is such that ${V_2}^{u(t)} = {V_2}^t \cup \{\overline{a}\}$ and ${V_1}^{u(t)} = {V_1}^t$. As V_1 is invariant, $u(t)$ trivially satisfies (8a), but it satisfies (8b) iff ${V_1}^t$ contains a tuple agreeing with $\overline{a}$ on the first two elements. In other words, we can insert $\overline{a}$ into ${V_2}^t$ only if there is a tuple $\langle e, d, p \rangle$, for some p, in the extension of V_1. This corresponds to condition (a) of Theorem 3 in [6] for the translatability of insertions, while condition (b) is necessary only if we assume that $\overline{a}$ does *not* belong to ${V_2}^t$, that is, the tuple we want to insert is not already present in the extension of V_2 before the update. Finally, checking that $u(t)$ satisfies all of the edg's in $\widetilde{\Sigma}_{\mathcal{V}}$, namely (8c), (8d) and (8e), corresponds to condition (c).

It should appear clear that with $\left|2^{\mathcal{V}}\right|$ choices for the complement symbols, stating conditions à la [6] for the translatability w.r.t. an instance for each possible view update when $|\mathcal{V}| > 2$ could be quite tedious. Fortunately such conditions are subsumed by our general criterion and, if needed, can be derived from it in each case.

Example 3. Let U and $\Sigma_{\mathcal{R}}$ as in Example 2, but let $\mathcal{V} = \{V_1, V_2, V_3\}$ and $\Sigma_{\mathcal{RV}}$ consist of the formulae defining V_1, V_2 and V_3 as projections on EDP, PS and DM, respectively. It is easy to verify that $\mathcal{V}$ is acyclic and that Σ implies the jd $\bowtie[EDP, PS, DM]$. By substituting the explicit definition of R in terms of $\mathcal{V}$ in Σ, we obtain $\widetilde{\Sigma}_{\mathcal{V}}$ consisting of the inclusion dependencies $V_1[D] \subseteq V_3[D]$, $V_3[D] \subseteq V_1[D]$, $V_1[P] \subseteq V_2[P]$ and $V_2[P] \subseteq V_1[P]$, and of the fd's $V_1\colon ED \to P$, $V_2\colon P \to S$ and $V_3\colon D \to M$.

Let $\mathcal{C} = \{V_3\}$. The update u_1 expressed by $\{\mathsf{insert}_{V_1}(\langle e,d,p\rangle), \mathsf{insert}_{V_2}(\langle p,s\rangle), \mathsf{noop}_{V_3}\}$ is translatable w.r.t. a legal view state t iff there is a tuple $\langle d,m\rangle$ in ${V_3}^t$ for some m (which corresponds to satisfying the ind $V_1[D] \subseteq V_3[D]$, while the other ones are trivially satisfied) and $u_1(t)$ satisfies all the fd's in $\widetilde{\Sigma}_{\mathcal{V}}$ involving V_1 and V_2. The view update u_2 expressed by $\{\mathsf{insert}_{V_1}(\langle e',d',p'\rangle), \mathsf{noop}_{V_2}, \mathsf{noop}_{V_3}\}$ is translatable w.r.t. t iff there are tuples $\langle p',s\rangle \in {V_2}^t$ and $\langle d',m\rangle \in {V_3}^t$ for some s and m, respectively, and $u_2(t)$ satisfies all the fd's in $\widetilde{\Sigma}_{\mathcal{V}}$ involving V_1.

Note that the view update u_1 in Example 3 requires the simultaneous insertion of tuples into the extension of both V_1 and V_2, which in practise (e.g., in SQL) would be achieved by means of a transaction consisting of two successive insertions.

Another difference between our general criterion for translatability (w.r.t. a view state) and the approach followed in [6] is that, while the latter requires some tests on the view instance and some other at the database level, the former can be checked entirely at the view level.

We conclude the section with a note about the problem of checking translatability of view updates w.r.t. *every* view state, and not just a given one. This is indeed the problem on which Bancilhon and Spyratos were originally focus in [2], but it is ignored in [6]. The characterisation we gave in our Theorem 6 provides a method that, even though possibly incomplete, allows to check whether a view update is translatable w.r.t. every view state. Apart from the trivial update consisting of noop_{V_i} for each $V_i \in \mathcal{V}$, a view update which is always translatable in Example 3 is expressed by:

$$\big\{\big(\exists x\, V_1(x,d,p) \to \mathsf{insert}_{V_1}(e,d,p)\big) \wedge \big(\nexists x\, V_1(x,d,p) \to \mathsf{noop}_{V_1}\big), \mathsf{noop}_{V_2}, \mathsf{noop}_{V_3}\big\}$$

that is, insert tuple $\langle e,d,p\rangle$ into the extension of V_1 only if there exists already another tuple with the same value for attributes Department and Position, otherwise do nothing.

5 Conclusion and Future Work

We presented a framework, based on the notion of view under constraints, which is an instance of B&S' abstract one, in that we consider only view mappings that are expressible by means of FOL constraints. By using the notion of definability, we gave a constructive characterisation of when and whether a view induced by a set of constraints is invertible, and we provided a general criterion, based on the idea of "embedding" of the constraints, for testing whether a FO-expressible view update is translatable. We studied an application setting, which extends the one considered in [6] and in which our framework is complete, and we compared our general criterion for translatability of updates w.r.t. an instance with the conditions given in [6] for insertions, deletions and replacements. Although our approach might not be suitable for every application setting, we believe that it can provide some guidance in a field which remains still largely unexplored.

For what concerns future work, the following directions seem worth of further investigation:

1. identify other fragments where implicit definability is finitely controllable (e.g., views defined by selections) and explore the potential of languages in the Datalog$^{\pm}$ family [5] in this sense;
2. in particular, further extend the setting presented in Sec. 4 to multiple database relations and views defined by projections over joins, allowing also some form of non-full tgd's as database constraints (possible candidates are acyclic inclusion dependencies and non-key-conflicting tgd's);
3. study the connection with logical abduction with respect to the possibility of finding view complements and (classes of) translatable updates.

We conjecture that implicit definability in the setting of point 2 is finitely controllable, but that the inverse mapping might not necessarily be given by the join operator in this case, hence the explicit definitions of each database symbol in terms of the view symbols should be obtained through rewriting techniques like the ones described, e.g., in [11,8].

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley (1995)
2. Bancilhon, F., Spyratos, N.: Update semantics of relational views. ACM Transactions on Database Systems 6(4), 557–575 (Dec 1981)
3. Beeri, C., Vardi, M.Y.: On acyclic database decompositions. Information and Control 61(2), 75–84 (1984)
4. Beth, E.W.: On Padoa's method in the theory of definition. Indagationes Mathematicae 15, 330–339 (1953)
5. Calì, A., Gottlob, G., Lukasiewicz, T.: Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In: Proceedings of the 12th International Conference on Database Theory. pp. 14–30. ICDT '09, ACM, New York, NY, USA (2009)
6. Cosmadakis, S.S., Papadimitriou, C.H.: Updates of relational views. Journal of the Association for Computing Machinery 31(4), 742–760 (Oct 1984)
7. Franconi, E., Guagliardo, P.: On the translatability of view updates. Tech. Rep. KRDB12-1, KRDB Research Centre, Free University of Bozen-Bolzano (Mar 2012), `http://www.inf.unibz.it/krdb/pub/TR/KRDB12-2.pdf`
8. Franconi, E., Kerhet, V., Ngo, N.: Exact query reformulation with expressive ontologies and databases. Tech. Rep. 12158, KRDB Tech research group, Free University of Bozen-Bolzano (Mar 2012), `http://www.inf.unibz.it/krdb/pub/TR/KRDB-Tech-12158.pdf`
9. Gottlob, G., Paolini, P., Zicari, R.: Properties and update semantics of consistent views. ACM Transactions on Database Systems 13(4), 486–524 (Dec 1988)
10. Gurevich, Y.: Toward logic tailored for computational complexity. In: Computation and Proof Theory, Lecture Notes in Mathematics, vol. 1104, pp. 175–216. Springer Berlin / Heidelberg (1984)
11. Huang, G.: Constructing Craig interpolation formulas. In: Computing and Combinatorics, Lecture Notes in Computer Science, vol. 959, pp. 181–190. Springer Berlin / Heidelberg (1995)
12. Lechtenbörger, J.: The impact of the constant complement approach towards view updating. In: Proceedings of PODS 2003. pp. 49–55. San Diego, CA (Jun 2003)