Freie Universität Bozen
Libera Università di Bolzano
Free University of Bozen · Bolzano

Faculty of Computer Science, Free University of Bozen-Bolzano
Piazza Domenicani 3, I-39100 Bolzano, Italy
Tel: +39 04710 16000, Fax: +39 04710 16009

KRDB Research Centre Technical Report:

On the translatability of view updates

Paolo Guagliardo, Enrico Franconi

Affiliation	KRDB Research Centre, Free University of Bozen-Bolzano
Corresponding author	guagliardo@inf.unibz.it
Keywords	views, updates, translatability, definability
Number	KRDB12-1
Date	March 12, 2012
URL	http://www.inf.unibz.it/krdb/tr/KRDB12-1.pdf

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 we compare our general criterium 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, rather than just two, and full dependencies (that is, egd's and full tgd's) as database constraints, rather than functional dependencies only.

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.

As an example of the problems that arise in view update, consider the situation shown in Figure 1, where the view V is defined as the natural join of the database relations E and D. The initial content of E, D and V consists only of those tuples having no mark on their left side. Suppose we update V by inserting the †-marked tuple ⟨Ford, Sales, Reyes⟩ into it. Then, in order to reflect this change also in the database, the †-marked tuples ⟨Ford, Sales⟩ and ⟨Sales, Reyes⟩ must be necessarily inserted into E and D, respectively. At this point, however, the new tuple ⟨Ford, Sales⟩ in E joins with ⟨Sales, Locke⟩ in D, resulting in ⟨Ford, Sales, Reyes⟩ being unexpectedly shown in V as a "side effect". Such update anomalies, introducing changes in the view not directly wanted nor explicitly made by the user, are clearly unacceptable.

$V = E \bowtie D$

	EMP	DEP	MGR
	Kwon	Tech	Hume
†	Ford	Sales	Reyes
‡	Ford	Sales	Locke

E

	EMP	DEP
	Kwon	Tech
†	Ford	Sales

D

	DEP	MGR
	Tech	Hume
	Sales	Reyes
†	Sales	Locke

Figure 1: The insertion of the †-marked tuple into the view V is propagated as the insertion of the †-marked tuples into E and D, causing the ‡-tuple to unexpectedly appear in V.

An additional difficulty is caused by the fact that there might be more than one possible database update corresponding to a given update on the view, and we cannot simply pick one because in most situations a "right" choice does not even exist. For instance, the insertion of the †-marked tuple ⟨Ford, Sales⟩ into the view V of Figure 2 could be propagated by inserting into the database relation R either of the two †-marked tuples ⟨Ford, Sales, Reyes⟩ and ⟨Ford, Sales, Austen⟩, or even both of them at the same time. As a matter of fact, we can pick anything for the MGR attribute in this particular case. Clearly, the presence of constraints (e.g., a functional dependency R: MGR $\rightarrow$ DEP) would restrict the values that can be used.

Another complication concerns updates that modify the database relations even though this is not strictly necessary in order to reflect the changes made on the view. As an example, consider once more the situation in Figure 2 (where R and V initially consists only of the unmarked rows) and the insertion of the tuple ⟨Kwon, Tech⟩ into V. Clearly, since V already contains that tuple,

R

	EMP	DEP	MGR
	Kwon	Tech	Hume
†	Ford	Sales	Reyes
†	Ford	Sales	Austen
‡	Kwon	Tech	Austen

$V = \pi_{\text{EMP,DEP}}(R)$

	EMP	DEP
	Kwon	Tech
†	Ford	Sales

Figure 2: The view V is the projection over the attributes EMP and DEP of the database relation R.

the view update in question has no effect. Nevertheless, we could think of propagating it as the insertion into R of a tuple like the ‡-marked one ⟨Kwon, Tech, Austen⟩, which after all reflects exactly the changes on the view (i.e., no change). However, R is modified without reason and this should be forbidden.

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 [11] 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. [8] 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 [7] 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, [7] 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.

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 [7] and generalise the results to a more general setting; we conclude in Sec. 5 by pointing out some future research directions.

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 (that is, if S is an n-placed relation symbol, then S^s is an n-ary relation). We assume the interpretation function $\cdot^s$ of a relational structure s over $\mathcal{S}$ with domain Δ to be from $\mathcal{S}$ to $\bigcup_{i \geq 1} \mathcal{P}(\Delta^i)$, where $\mathcal{P}(A)$ denotes the powerset of A. Given a relational structure s over $\mathcal{S}$, its *restriction* to a subset $\mathcal{S}'$ of $\mathcal{S}$ is the relational structure $s|_{\mathcal{S}'}$ obtained from s by restricting its interpretation function to the relation symbols in $\mathcal{S}'$. 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 $\overline{x}$.

Given two disjoint signatures $\mathcal{S}$ and $\mathcal{S}'$ and a finite set Σ of constraints over $\mathcal{S} \cup \mathcal{S}'$, we say that a relation symbol $S \in \mathcal{S}$ is *implicitly defined* by the relation symbols in $\mathcal{S}'$ under Σ if and only if, for every two Σ-models s and t such that $\Delta^s = \Delta^t$, we have that $S^s = S^t$ whenever $s|_{\mathcal{S}'} = t|_{\mathcal{S}'}$. In other words, $S \in \mathcal{S}$ is implicitly definable from $\mathcal{S}'$ if any two models of Σ that have the same domain and agree in what they assign to the symbols in $\mathcal{S}'$ also agree in what they assign to S, that is, the extension of S depends only on the extension of the symbols in $\mathcal{S}'$. We say that $S \in \mathcal{S}$ is *explicitly defined* by the relation symbols in $\mathcal{S}'$ under Σ iff there is a formula $\phi_S(\overline{x})$, with as many free variables as the arity of S, such that $\Sigma \models \forall \overline{x}\big(S(\overline{x}) \leftrightarrow \phi_S(\overline{x})\big)$ and $\mathsf{sig}\big(\phi_S(\overline{x})\big) \subseteq \mathcal{S}'$, called an *explicit definition* of S with respect to $\mathcal{S}'$ under Σ.

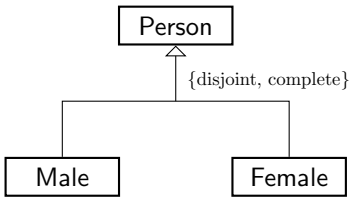

Figure 3: A UML class diagram where each class is implicitly defined by the others (e.g., knowing all the persons and who the males are, we also implicitly know who the females are).

Example 1. Consider the UML class diagram depicted in Figure 3, 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. ■

Clearly, if a relation symbol $S \in \mathcal{S}$ has an explicit definition w.r.t. $\mathcal{S}'$ under constraints Σ, then S is also implicitly defined by $\mathcal{S}'$ under Σ. For instance, if $A(x)$ has the explicit definition $B(x) \vee C(x)$, that is, $\Sigma \models \forall x\,.\,A(x) \leftrightarrow \big(B(x) \vee C(x)\big)$, then the interpretation of A directly depends on what is assigned to B and C. In other words, explicit definability always implies implicit definability. The converse is not true in general, that is, knowing that a symbol is implicitly defined by some other ones does not mean that we can actually find an explicit definition of it in terms of those symbols. The following fundamental result due to Beth [4] establishes that this is actually the case for first-order logic.

Theorem (Beth's Definability). *Let $\mathcal{S}$ and $\mathcal{S}'$ be disjoint signatures, and Σ be a finite set of first-order logic constraints over $\mathcal{S} \cup \mathcal{S}'$. If $S \in \mathcal{S}$ is implicitly definable from $\mathcal{S}'$ under Σ, then S has an explicit definition with respect to $\mathcal{S}'$ under Σ.*

A *renaming* over a signature $\mathcal{S}$ is a bijective function $\mathsf{ren} : \mathcal{S} \rightarrow \mathcal{S}'$, 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 $\mathcal{R} \cup \mathcal{V}$. We call *database constraints* the set $\Sigma_{\mathcal{R}} = \{\phi \in \Sigma \mid \mathsf{sig}(\phi) \subseteq \mathcal{R}\}$, *view constraints* the set $\Sigma_{\mathcal{V}} = \{\phi \in \Sigma \mid \mathsf{sig}(\phi) \subseteq \mathcal{V}\}$ and *inter-schema constraints* all other formulae in Σ that are not database nor view constraints. Obviously, since $\mathcal{R}$ and $\mathcal{V}$ are disjoint with each other, the database, view and inter-schema constraints form a partition of the global constraints Σ. Note also that, in general, inter-schema constraints need not provide the definitions of the view symbols explicitly (this is quite common in practise, though). For instance, consider once again Example 1 and take $\{\mathsf{Person}, \mathsf{Male}\}$ as database schema and $\{\mathsf{Female}\}$ as view schema; then, all of the formulae in Σ are inter-schema constraints (each mentions symbols from both signatures), but none of them directly provides an explicit definition for Female, which is nevertheless defined implicitly.

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., $\mathbf{T}_\Sigma$).

Our framework is based on the central notion of *view under constraints*, which naturally extends with explicit constraints the definition of view used in [2]. Indeed, when only database constraints are present, the two notions essentially coincide, although in [2] constraints (at the database level) are considered only in an implicit way.

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

We write $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ to indicate that every $V \in \mathcal{V}$ is implicitly definable from $\mathcal{R}$ under constraints Σ. In addition, 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. 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. Therefore, we introduce definability in terms of states as follows.

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

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 exists one and only one view from $\mathcal{R}$ to $\mathcal{V}$ under Σ.*

Proof. We will show that there exist two distinct views from $\mathcal{R}$ to $\mathcal{V}$ under Σ if and only if $\mathcal{R} \not\twoheadrightarrow_\Sigma \mathcal{V}$.

"if". Assume $\mathcal{R} \not\twoheadrightarrow_\Sigma \mathcal{V}$, then for some $s \in \mathbf{S}_\Sigma$ there are $t', t'' \in \mathbf{T}_\Sigma$ with $t' \neq t''$ such that $s \uplus t'$ and $s \uplus t''$ are models of Σ. Therefore, we can construct two views f' and f'' such that $f'(s) = t'$ and $f''(s) = t''$.

"only if". Let f and f' be views from $\mathcal{R}$ to $\mathcal{V}$ under Σ such that $f(s) \neq f'(s)$ for some $s \in \mathbf{S}_\Sigma$. Then, both $s \uplus f(s)$ and $s \uplus f'(s)$ are models of Σ, hence $\mathcal{R} \not\twoheadrightarrow_\Sigma \mathcal{V}$. □

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 when something is implicitly defined it is possible to find its explicit definition in FOL. Unfortunately, Beth's result [4] holds when instances are *unrestricted* (i.e., allowed to be possibly infinite), while it fails [9] 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 discuss an application setting in which implicitly definability is finitely controllable in Sec. 4.

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.

Definition 3 (Strict update). Let $f\colon \mathbf{S}_\Sigma \to \mathbf{T}_\Sigma$ be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ. An update $u \in U_\mathcal{V}$ is called *strict on* f iff there exists $t \in f(\mathbf{S}_\Sigma)$ such that $u(t) \neq t$. The set of all the updates that are strict on f is denoted by U_f. ■

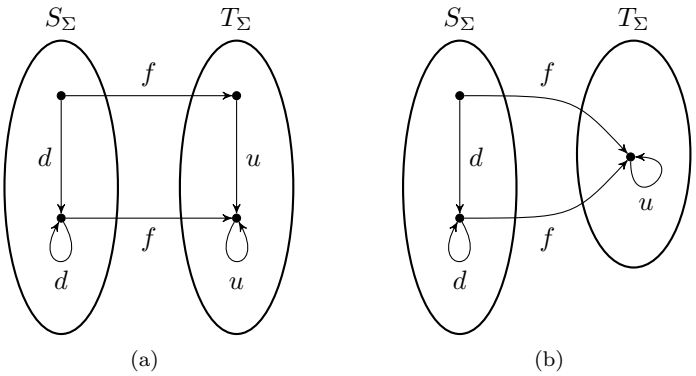

Figure 4: (a) The database update d is a translation of the view update u. (b) The database update d is consistent with the view update u but not acceptable.

In other words, a strict update w.r.t. a view f is one that does not coincide with the identity mapping on the image of f.

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. These requirements are formalised below (cf. Definition 3.1 in [2]) and exemplified in Figure 4a.

DEFINITION 4 (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 (*consistent*)
(2) $\forall s \in \mathbf{S}_\Sigma,\ uf(s) = f(s) \implies d(s) = s\,.$ (*acceptable*) ■

A situation like the one shown in Figure 4b, where we exhibit a database update that is consistent with a given view update but not acceptable, cannot happen if the view mapping is injective.

PROPOSITION 1. *Let f be an injective view from $\mathcal{R}$ to $\mathcal{V}$ under Σ and let $u \in U_\mathcal{V}$. Then, every database update consistent with u is acceptable.*

Proof. Let $d \in U_\mathcal{R}$. We need to show that the first condition of Definition 4 implies the second one. Thus, let $s \in \mathbf{S}_\Sigma$ and assume $uf(s) = fd(s)$. Now, if $uf(s) = f(s)$, we get $fd(s) = f(s)$ which, by the injectivity of f, implies $d(s) = s$. □

A translation of a given update on a view f 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, which is what Definition 4 indeed requires. Therefore, before we start looking for a translation,

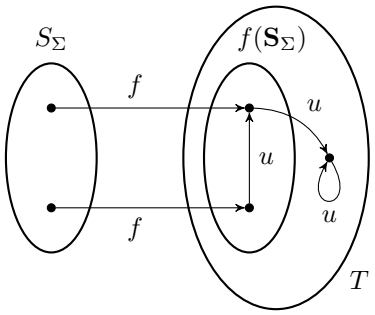

Figure 5: The view update u is not translatable, as it leads to a view state outside the image of f, therefore not reachable from any database state.

we first need to make sure that the given view update allows for one. A view update that can be translated into a suitable database update, that is, for which there exists a translation, is called *translatable*. We give an example of untranslatable update in Figure 5.

Definition 5 (Translatability). Let $f\colon \mathbf{S}_\Sigma \to \mathbf{T}_\Sigma$ 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 $s \in \mathbf{S}_\Sigma$ there exists $s' \in \mathbf{S}_\Sigma$ such that $f(s') = uf(s)$. ■

Note that the condition of translatability given in Definition 5 is equivalent to saying that u is translatable if and only if $u\big(f(\mathbf{S}_\Sigma)\big) \subseteq f(\mathbf{S}_\Sigma)$. It is also easy to verify that, as expected, a view update is translatable if and only if there exists a translation of it.

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.

One of the factors contributing to the existence of multiple translations is the loss of information that occurs when two distinct database states collapse into the same view state, that is, when the view mapping is not injective. Intuitively, if a view update results in a view state that is the image of two different database states, we then have two different alternatives for "going back" to the database, and therefore multiple ways of translating the update. However, lack of injectivity in the view is not sufficient for a translatable view update to admit more than one translation. In fact, there are uniquely translatable updates on views that are not injective, as witnessed by the example in Figure 6a. Indeed, even in the presence of view states corresponding to more than one database state, there is only one possible translation (if one at all) as long as the view update does not result in one of these "ambiguous" view states. This is stated more precisely in the following lemma.

Lemma 1. *Let f be a view under Σ and let $u \in U_\mathcal{V}$. Denote by $\boldsymbol{S}^f_\Sigma$ the set of all states $s \in \boldsymbol{S}_\Sigma$ for which there does not exist $s' \in \boldsymbol{S}_\Sigma$ such that $s \neq s'$ and $f(s) = f(s')$. Then, u is uniquely translatable if and only if $u\big(f(\boldsymbol{S}_\Sigma)\big) \subseteq f(\boldsymbol{S}^f_\Sigma)$.*

Proof. We show that $u\big(f(\mathbf{S}_\Sigma)\big) \not\subseteq f(\mathbf{S}^f_\Sigma)$ if and only if u is not uniquely translatable.

"if". Assume that u is not uniquely translatable. If u is not translatable at all, $u\big(f(\mathbf{S}_\Sigma)\big) \not\subseteq f(\mathbf{S}_\Sigma)$ and, since $f(\mathbf{S}^f_\Sigma) \subseteq f(\mathbf{S}_\Sigma)$, we get $u\big(f(\mathbf{S}_\Sigma)\big) \not\subseteq f(\mathbf{S}^f_\Sigma)$. Let us now consider the case in which u is translatable, but there exist two translations d_1 and d_2 of u such that $d_1(s) \neq d_2(s)$ for some $s \in \mathbf{S}_\Sigma$. Then, as d_1 and d_2 are both translations of u, we have $fd_1(s) = uf(s) = fd_2(s)$. Therefore, $d_1(s)$ and $d_2(s)$ are not in $\mathbf{S}^f_\Sigma$ and $uf(s) \notin f(\mathbf{S}^f_\Sigma)$.

"only if". Assume that $u\big(f(\mathbf{S}_\Sigma)\big) \not\subseteq f(\mathbf{S}^f_\Sigma)$. Then, there is a state $t \in f(\mathbf{S}_\Sigma)$ such that $u(t) \notin f(\mathbf{S}^f_\Sigma)$. Moreover, as t is in the image of f, there exists $s \in \mathbf{S}_\Sigma$ for which $f(s) = t$. If $u(t) \notin f(\mathbf{S}_\Sigma)$, then u is not translatable at all, and we are done. Thus, let us assume $u(t) \in f(\mathbf{S}_\Sigma)$. Then, there exists $s' \in \mathbf{S}_\Sigma$ such that $f(s') = u(t)$. This means that there is a translation d_1 of u for which $d_1(s) = s'$. Since $u(t) \notin f(\mathbf{S}^f_\Sigma)$, there must be some state $s'' \in \mathbf{S}_\Sigma$ such that $s'' \neq s'$ and $f(s'') = f(s')$. Hence, there exists also a translation d_2 of u for which $d_2(s) = s''$. Therefore, since $d_1 \neq d_2$, we conclude that u is not uniquely translatable. □

Clearly, for an injective view, $\mathbf{S}^f_\Sigma$ coincides with $\mathbf{S}_\Sigma$, therefore a view update is translatable if and only if it is uniquely translatable. The following theorem gives a characterisation of the unique database update into which a translatable view update can be translated when the view mapping 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 $d = \hat{f}^{-1}\hat{u}\hat{f}$.*

Proof. Special case of Theorem 5.5 and Theorem 5.6 in [2] when $g \equiv 0$. Below, however, we provide a detailed proof.

As f is injective, $\hat{f}$ is a bijection from $\mathbf{S}_\Sigma$ to $f(\mathbf{S}_\Sigma)$, hence invertible. Moreover, by Proposition 1, it is enough to show that only the first condition of Definition 4 is satisfied. For every $s \in \mathbf{S}_\Sigma$, we have the following:

$$
\begin{aligned}
fd(s) &= f\hat{f}^{-1}\hat{u}\hat{f}(s) = f\hat{f}^{-1}\hat{u}\big(\hat{f}(s)\big) \\
&= f\hat{f}^{-1}\hat{u}\big(f(s)\big) && \big[\text{by def. of } \hat{f}\text{, since } s \in \mathbf{S}_\Sigma\big] \\
&= f\hat{f}^{-1}\big(\hat{u}(t)\big) && \big[\text{by taking } t = f(s)\big] \\
&= f\hat{f}^{-1}\big(u(t)\big) && \big[\text{by def. of } \hat{u}\text{, since } t \in f(\mathbf{S}_\Sigma)\big] \\
&= f\big(\hat{f}^{-1}(s')\big) && \big[\text{by taking } s' = u(t)\big] \\
&= \hat{f}\big(\hat{f}^{-1}(s')\big) && \big[\text{by def. of } \hat{f}\text{, as } \hat{f}^{-1}(s') \in \mathbf{S}_\Sigma\big] \\
&= s' = u(t) = uf(s) && \square
\end{aligned}
$$

As already noted, a view update may be uniquely translatable even when the view is not injective. However, a characterisation of the translation, in the same spirit of the one given above for injective views, though possible, becomes more involved in this case. In this article, we do not pursue that direction, but rather focus on enforcing injectivity.

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

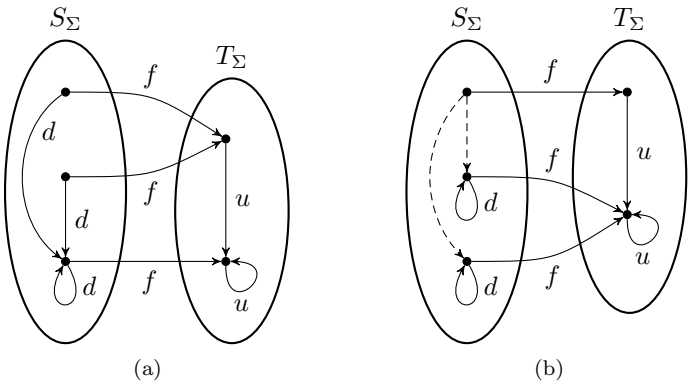

Figure 6: (a) Although f is not injective, d is the only possible translation of the view update u. (b) The dashed paths are mutually exclusive alternatives for d, resulting in different translations of u.

3.1 Invertibility of Views

In order to apply the result of Theorem 2, we are required to know, in the first place, whether a view is injective. More importantly, once in the presence of an injective view mapping, we also need some way of computing the inverse of its surjective restriction, so as to effectually obtain the unique translation of any translatable view update. In B&S' framework, view mappings are generic functions and no attempt is made in providing computational algorithms for them and their inverses; therefore, the translation of a uniquely translatable update is characterised only in abstract functional terms, that is, as given in Theorem 2. In our setting, however, view mappings are functions induced (i.e., expressible) by sets of constraints, and a constructive characterisation of their inverses is possible, as we shall see, by means of logical definability.

A sufficient condition for the injectivity of a view under constraints is that each of the database symbols be implicitly definable under the given constraints in terms of the view symbols. In addition, the same condition ensures that the function induced by the constraints, opportunely restricted on the appropriate domain, is the inverse of the surjection induced by the view.

LEMMA 2. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, and let $\mathcal{V} \twoheadrightarrow^{h}_{\Sigma} \mathcal{R}$. Then, 1) f is injective; and 2) the restriction of h to the image of f is the inverse of the surjection induced by f.*

Proof.

1) Suppose f is not injective, that is, there are states $s, s' \in \mathbf{S}_\Sigma$ such that $s \neq s'$ and $f(s) = f(s')$. But then, since f is a view under Σ, $s \uplus f(s)$ and $s' \uplus f(s')$ are both models of Σ, in contradiction of $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$.
2) Let $\hat{f}\colon \mathbf{S}_\Sigma \to f(\mathbf{S}_\Sigma)$ be the surjection induced by f and let $\hat{h}\colon f(\mathbf{S}_\Sigma) \to \mathbf{S}_\Sigma$ be the restriction of h to $f(\mathbf{S}_\Sigma)$. Clearly, since f is injective, $\hat{f}$ is a bijection, hence invertible. As $\hat{f}(s) = f(s)$ for every $s \in \mathbf{S}_\Sigma$ and $\hat{h}(t) = h(t)$ for every $t \in f(\mathbf{S}_\Sigma)$, proving that $\hat{h} = \hat{f}^{-1}$ amounts to showing that $hf(s) = s$ for every $s \in \mathbf{S}_\Sigma$. Thus, let $s \in \mathbf{S}_\Sigma$ and $t = f(s)$. Since f is a view under Σ,

$s \uplus t$ is a model of Σ. Let $s' = h(t)$ and suppose that $s' \neq s$. As h is a view under Σ, $s' \uplus t$ is also a model of Σ, in contradiction of $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$. Therefore, $s = s' = h(t) = hf(s)$. □

Note that, in the above lemma, no assumption is made on the view itself, that is, f can be any view under constraints. Next, we consider the case when the view is induced by the constraints, and study its injectivity and surjectivity.

LEMMA 3. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$. Then, 1) f is surjective; and 2) f is injective if and only if $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$.*

Proof.

1) Suppose f is not surjective, that is, there are states $s \in \mathbf{S}_\Sigma$ and $t \in \mathbf{T}_\Sigma$ such that $s \uplus t \models \Sigma$ and $t \neq f(s)$. Since f is a view under Σ, $s \uplus f(s) \models \Sigma$ too, in contradiction of $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$.
2) By Lemma 2, f is injective whenever $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$, therefore we just need to show the "only if" direction and we do so by contraposition. Assume $\mathcal{V} \not\twoheadrightarrow_\Sigma \mathcal{R}$; then, there exist Σ-models $s \uplus t$ and $s' \uplus t$ with $s \neq s'$. Since $s, s' \in \mathbf{S}_\Sigma$ and f is a view under Σ, $s \uplus f(s)$ and $s' \uplus f(s')$ are also models of Σ. But then, as $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$, we have that $f(s) = t = f(s')$. Therefore, f is not injective. □

In typical scenarios, a view mapping is usually specified by providing an explicit definition of each view symbol in terms of the database symbols, and it is therefore trivially induced by the constraints.

The following is a direct important consequence of Lemma 2 and Lemma 3, establishing that 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.

THEOREM 3. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$. Then, f is invertible if and only if $\mathcal{V} \twoheadrightarrow^h_\Sigma \mathcal{R}$, and in such a case $h = f^{-1}$.* □

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

$$\Sigma \cup \mathsf{ren}(\Sigma) \models \forall \overline{x} \left(R(\overline{x}) \leftrightarrow R'(\overline{x}) \right) \ , \tag{5}$$

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 (5). That is, $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ can be reformulated as follows:

$$\Sigma \cup \mathsf{ren}(\Sigma) \models \forall \overline{x} \left(R(\overline{x}) \rightarrow R'(\overline{x}) \right) \ , \tag{6}$$

which is equivalent to the problem of checking containment of (atomic) queries under constraints.

Given a view f from $\mathcal{R}$ to $\mathcal{V}$ induced by a set of constraints Σ over $\mathcal{R} \cup \mathcal{V}$, we define the problem $\mathsf{invert}(\mathcal{V}, \Sigma)$ of determining whether f is invertible, which amounts to check whether $\mathsf{impl\text{-}def}(R, \mathcal{V}, \Sigma)$ for each $R \in \mathcal{R}$ (where $\mathcal{R}$ is given by $\mathsf{sig}(\Sigma) \setminus \mathcal{V}$).

3.2 Translatability of View Updates

In this section, we provide an interesting characterisation of when a view update is translatable. The general idea consists in imposing additional constraints on the view schema so that every *legal* view update is translatable and vice versa. Indeed, we will show that under certain conditions it is possible to find a set of view constraints such that a view update satisfies them if and only if it is translatable.

A consistent set of constraints over $\mathcal{R} \cup \mathcal{V}$ is *$\mathcal{R}$-defining* iff it consists exactly of one formula for each $R \in \mathcal{R}$ of the form $\forall \overline{x}\,\big(R(\overline{x}) \leftrightarrow \phi_R(\overline{x})\big)$, with $\mathsf{sig}\big(\phi_R(\overline{x})\big) \subseteq \mathcal{V}$. For a finite satisfiable set of global constraints Σ and an $\mathcal{R}$-defining set Θ, we denote by $\Theta(\Sigma)$ the set of view constraints obtained by replacing, for each $R \in \mathcal{R}$, every occurrence of $R(\overline{x})$ in Σ with the definition $\phi_R(\overline{x})$ given in Θ. Clearly, an $\mathcal{R}$-defining set is such that $\mathcal{V} \twoheadrightarrow_\Theta \mathcal{R}$ and induces a function θ from $\mathcal{V}$ to $\mathcal{R}$. Since Θ does not contain nor entail any database or view constraints, every view state $t \in \mathbf{T}$ is Θ-consistent and therefore in the domain of θ.

Lemma 4. *Let Θ be an $\mathcal{R}$-defining set of constraints and let θ be the view from $\mathcal{V}$ to $\mathcal{R}$ induced by Θ. Let Σ be a finite satisfiable set of global constraints such that $\Sigma \models \Theta$, and let $t \in \boldsymbol{T}$. Then, $\theta(t) \uplus t \models \Sigma$ if and only if $t \models \Theta(\Sigma)$.*

Proof. As Θ is $\mathcal{R}$-defining and θ is induced by Θ, for every $t \in \mathbf{T}$ we have that $\theta(t) \uplus t \models \Theta$ and, for every $\psi \in \Sigma$, that

$$\theta(t) \uplus t \models \forall \overline{x}\,\big(\psi(\overline{x}) \equiv \psi'(\overline{x})\big)\ , \qquad (*)$$

where ψ' is obtained by replacing every occurrence in ψ of each database predicate with the corresponding definition given in Θ. Then, we have the following:

$$\begin{aligned}
&\theta(t) \uplus t \models \Sigma \iff \forall \psi \in \Sigma,\ \theta(t) \uplus t \models \psi \\
&\quad\iff \forall \psi \in \Sigma,\ \theta(t) \uplus t \models \psi' && \big[\text{by } (*)\big] \\
&\quad\iff \theta(t) \uplus t \models \Theta(\Sigma) && \big[\text{by definition of } \Theta(\Sigma)\big] \\
&\quad\iff t \models \Theta(\Sigma) && \big[\text{since } \mathsf{sig}\big(\Theta(\Sigma)\big) \subseteq \mathcal{V}\big].\ \square
\end{aligned}$$

We know by Beth's theorem that whenever $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$ there exists an explicit definition for each of the database symbols in terms of the view symbols, that is, the constraints entail an $\mathcal{R}$-defining set Θ. In such a case, we say that $\Theta(\Sigma)$ "embeds" Σ and denote it by $\widetilde{\Sigma}_\mathcal{V}$, which we call the *$\mathcal{V}$-embedding* of Σ. From Lemma 4, we then directly get the following.

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

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 $t \in \boldsymbol{T}_\Sigma$.*

Proof. Since f is surjective, $f(\mathbf{S}_\Sigma) = \mathbf{T}_\Sigma$. Hence, u is translatable iff $u(\mathbf{T}_\Sigma) \subseteq \mathbf{T}_\Sigma$, that is, iff $u(t) \in \mathbf{T}_\Sigma$ for every $t \in \mathbf{T}_\Sigma$ and, as $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$, by Corollary 4 we have that $u(t) \in \mathbf{T}_\Sigma$ iff $u(t) \models \widetilde{\Sigma}_\mathcal{V}$. □

Note that, under the assumptions of Theorem 5, every globally consistent view state is in the image of the view and, in addition, satisfies the $\mathcal{V}$-embedding of the global constraints. Thus, the above result essentially says that we need to make sure that, when updating a view state that is legal w.r.t. the embedded constraints, 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 $\mathcal{V}'$-defining set.

Insertion. For each view symbol $V \in \mathcal{V}$, the insertion of an arbitrary tuple $\overline{x}$ into (the extension of) V is represented by the following open formula:

$$\mathsf{insert}_V(\overline{x}) \stackrel{\text{def}}{\equiv} \forall \overline{y}\, \big[V'(\overline{y}) \leftrightarrow \big(V(y) \vee \overline{y} = \overline{x}\big)\big] \ . \tag{7}$$

Clearly, for every interpretation $s = \langle \Delta^s, \cdot^s \rangle$ over $\{V, V'\}$ and every assignment $\alpha\colon \overline{x} \mapsto \overline{a}$, where $\overline{a}$ is a tuple of values from Δ^s, we have that $s, \alpha \models \mathsf{insert}_V(\overline{x})$ iff $V'^s = V^s \cup \{\overline{a}\}$. Note that (7) can equivalently also be written as follows:

$$\forall \overline{y}\, \Big[\big(V(\overline{y}) \to V'(\overline{y})\big) \wedge \big(V'(\overline{y}) \to \big[V(\overline{y}) \vee \overline{y} = \overline{x}\,\big]\big) \wedge V'(\overline{x})\Big] \ . \tag{8}$$

Deletion. For each view symbol $V \in \mathcal{V}$, the deletion of an arbitrary tuple $\overline{x}$ from (the extension of) V is represented by the following open formula:

$$\mathsf{delete}_V(\overline{x}) \stackrel{\text{def}}{\equiv} \forall \overline{y}\, \Big[\big(V'(\overline{y}) \to V(\overline{y})\big) \wedge \big(V(\overline{y}) \to \big[V'(\overline{y}) \vee \overline{y} = \overline{x}\,\big]\big) \wedge \neg V'(\overline{x})\Big] \ . \tag{9}$$

For every interpretation $s = \langle \Delta^s, \cdot^s \rangle$ over $\{V, V'\}$ and every assignment $\alpha\colon \overline{x} \mapsto \overline{a}$, where $\overline{a}$ is a tuple of values from Δ^s, we have that $s, \alpha \models \mathsf{delete}_V(\overline{x})$ if and only if $V'^s = V^s \setminus \{\overline{a}\}$. Indeed, $s, \alpha \models \mathsf{delete}_V(\overline{x})$ iff $s \models \mathsf{delete}_V(\overline{a})$, that is, iff all of the conjuncts of (9) are satisfied by s after substituting $\overline{x}$ with $\overline{a}$, which gives:

$$V'^s \subseteq V^s \tag{10a}$$

$$V^s \subseteq V'^s \cup \{\overline{a}\} \tag{10b}$$

$$\overline{a} \notin V'^s \tag{10c}$$

Then, (10a) and (10c) together are equivalent to $V'^s \subseteq V^s \setminus \{\overline{a}\}$, while (10b) and (10c) give the other inclusion.

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

No change. For each view symbol $V \in \mathcal{V}$, any update that does not modify V can be represented by the following closed formula:

$$\mathsf{noop}_V \stackrel{\text{def}}{\equiv} \forall \overline{x}\, \big(V'(\overline{x}) \leftrightarrow V(\overline{x})\big)\ . \tag{11}$$

Replacement. For each view symbol $V \in \mathcal{V}$, the replacement in V of a tuple $\overline{x}$ with a tuple $\overline{y}$ is expressed by the following open formula:

$$\begin{aligned}\mathsf{replace}_V(\overline{x},\overline{y}) \stackrel{\text{def}}{\equiv} \Big[\big(\neg V(\overline{x}) \vee \overline{x} = \overline{y}\big) \rightarrow \mathsf{noop}_V\Big] \wedge \Big[\big(V(\overline{x}) \wedge \overline{x} \neq \overline{y}\big) \\ \rightarrow\ \forall \overline{z}\, \Big(\big(V'(\overline{z}) \rightarrow \big[V(\overline{z}) \vee \overline{z} = \overline{y}\big]\big) \wedge \big(V(\overline{z}) \rightarrow \big[V'(\overline{z}) \vee \overline{z} = \overline{x}\big]\big) \\ \wedge\ \neg V'(\overline{x}) \wedge V'(\overline{y})\Big)\Big]\ .\end{aligned} \tag{12}$$

Let $s = \langle \Delta^s, \cdot^s \rangle$ be an interpretation over $\{V, V'\}$ and let $\alpha\colon \{\, \overline{x} \mapsto \overline{a}, \overline{y} \mapsto \overline{b}\,\}$ be an assignment of the variable in $\overline{x}$ and $\overline{y}$, where $\overline{a}$ and $\overline{b}$ are tuples of values from Δ^s. We show that the semantics of $\mathsf{replace}_V(\overline{x},\overline{y})$ is the expected one for every such s and α. Indeed, we have that $s, \alpha \models \mathsf{replace}_V(\overline{x},\overline{y})$ iff $s \models \mathsf{replace}_V\big(\overline{a},\overline{b}\big)$, that is, iff the r.h.s. of (12) is satisfied by s after substituting $\overline{x}$ with $\overline{a}$ and $\overline{y}$ with $\overline{b}$. The antecedents of the two outermost implications in the r.h.s. of (12) define two mutually exclusive cases. When $\overline{a} = \overline{b}$ or $\overline{a} \notin V^s$, we have $V'^s = V^s$, that is, replacing a tuple with the same one or trying to replace a tuple which is not present in the extension of V has no effect. Otherwise, all of the following are true:

$$\overline{a} \in V^s \tag{13a}$$

$$\overline{a} \neq \overline{b} \tag{13b}$$

$$V'^s \subseteq V^s \cup \big\{\overline{b}\big\} \tag{13c}$$

$$V^s \subseteq V'^s \cup \{\overline{a}\} \tag{13d}$$

$$\overline{a} \notin V'^s \tag{13e}$$

$$\overline{b} \in V'^s \tag{13f}$$

Then, (13d) and (13e) give $V^s \setminus \{\overline{a}\} \subseteq V'^s$, from which we get $\big(V^s \setminus \big\{\overline{a}\big\}\big) \cup \big\{\overline{b}\big\} \subseteq V'^s$ using (13f), while (13c) and (13e) give $V'^s \subseteq \big(V^s \cup \big\{\overline{b}\big\}\big) \setminus \big\{\overline{a}\big\}$. Therefore, using (13b), we obtain $V'^s = \big(V^s \cup \big\{\overline{b}\big\}\big) \setminus \big\{\overline{a}\big\} = \big(V^s \setminus \big\{\overline{a}\big\}\big) \cup \big\{\overline{b}\big\}$, that is, replacing an existing tuple with a different one is the result of inserting the new tuple and then deleting the old one or, equivalently, deleting the old tuple and then inserting the new one.

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 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 if and only if $\widetilde{\Sigma}_\mathcal{V} \cup \Xi \models \mathsf{ren}(\widetilde{\Sigma}_\mathcal{V})$.*

Under the assumptions of the above theorem, the view f is injective by Lemma 2. 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 this section with the proof of Theorem 6, for which we first need to prove a general technical lemma that will also be used later on.

LEMMA 5. *Let Σ and Γ be finite sets of constraints over σ and γ, respectively, and let s be a relational structure over $\sigma \cup \gamma$. Then, $s \models \Sigma \cup \Gamma$ iff $s = t \uplus t' \uplus t''$, where t, t' and t'' are relational structures over $\sigma \setminus \gamma$, $\sigma \cap \gamma$ and $\gamma \setminus \sigma$, respectively, such that $t \uplus t' \models \Sigma$ and $t' \uplus t'' \models \Gamma$.*

Proof (of Lemma 5). As $\sigma \cup \gamma$ can be partitioned into $\sigma \setminus \gamma$, $\sigma \cap \gamma$ and $\gamma \setminus \sigma$, every relational structure over $\sigma \cup \gamma$ has the form $s = t \uplus t' \uplus t''$, where t, t' and t'' are relational structures with the same domain over such partitions.

"if". Assume $t \uplus t' \models \Sigma$ and $t' \uplus t'' \models \Gamma$, and suppose that $s \not\models \Sigma \cup \Gamma$. Then, there is a formula $\varphi \in \Sigma \cup \Gamma$ such that $t \uplus t' \uplus t'' \not\models \varphi$. If $\varphi \in \Sigma$, we have that $\mathsf{sig}(\varphi) \subseteq \sigma$ and, since $\mathsf{sig}(t'') \subseteq \gamma \setminus \sigma$, we get $t \uplus t' \not\models \varphi$, in contradiction of $t \uplus t' \models \Sigma$. Similarly, when $\varphi \in \Gamma$, we have that $\mathsf{sig}(\varphi) \subseteq \gamma$ and, as $\mathsf{sig}(t) \subseteq \sigma \setminus \gamma$, we obtain $t' \uplus t'' \not\models \varphi$, in contradiction of $t' \uplus t'' \models \Gamma$.

"only if". Assume $s \models \Sigma \cup \Gamma$. Then, $s \models \Sigma$ and $s \models \Gamma$. From $t \uplus t' \uplus t'' \models \Sigma$, as $\mathsf{sig}(\Sigma) \subseteq \sigma$ and $\mathsf{sig}(t'') \subseteq \gamma \setminus \sigma$, we obtain $t \uplus t' \models \Sigma$. Similarly, from $t \uplus t' \uplus t'' \models \Gamma$, as $\mathsf{sig}(\Gamma) \subseteq \gamma$ and $\mathsf{sig}(t) \subseteq \sigma \setminus \gamma$, we get $t' \uplus t'' \models \Gamma$. □

Proof (of Theorem 6). Let $\mathcal{V}'$ denote $\mathsf{ren}(\mathcal{V})$. By Lemma 5, a relational structure s is a model of $\widetilde{\Sigma}_\mathcal{V} \cup \Xi$ iff $s = t \uplus t'$, where t and t' are states over $\mathcal{V}$ and $\mathcal{V}'$, respectively, such that $t \models \widetilde{\Sigma}_\mathcal{V}$ and $t \uplus t' \models \Xi$. Since $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$, by Corollary 4 we have that $t \in \mathbf{T}_\Sigma$ iff $t \models \widetilde{\Sigma}_\mathcal{V}$. Moreover, denoting by ξ the function induced

by Ξ, we have that $t \uplus t' \models \Xi$ iff $t' = \xi(t)$. Hence, $t \uplus \xi(t) \models \widetilde{\Sigma}_{\mathcal{V}} \cup \Xi$ for every $t \in \mathbf{T}_{\Sigma}$ and, vice versa, every model of $\widetilde{\Sigma}_{\mathcal{V}} \cup \Xi$ has the form $t \uplus \xi(t)$ for some $t \in \mathbf{T}_{\Sigma}$. Then,

$$\begin{aligned}
&\widetilde{\Sigma}_{\mathcal{V}} \cup \Xi \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big) \\
&\quad \iff \forall s,\ s \models \widetilde{\Sigma}_{\mathcal{V}} \cup \Xi \ \text{ implies } \ s \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big) \\
&\quad \iff \forall t \in \mathbf{T}_{\Sigma},\ t \uplus \xi(t) \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big) \quad ;
\end{aligned}$$

and, for every $t \in \mathbf{T}_{\Sigma}$, we have that:

$$\begin{aligned}
&t \uplus \xi(t) \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big) \\
&\quad \iff \xi(t) \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big) && \big[\ \text{since } \mathsf{sig}\big(\mathsf{ren}(\widetilde{\Sigma}_{\mathcal{V}})\big) \subseteq \mathcal{V}', \\
&&& \quad \mathsf{sig}(t) \subseteq \mathcal{V},\ \mathcal{V} \cap \mathcal{V}' = \varnothing\ \big] \\
&\quad \iff \mathsf{ren}^{-1}\big(\xi(t)\big) \models \widetilde{\Sigma}_{\mathcal{V}} \\
&\quad \iff u(t) \models \widetilde{\Sigma}_{\mathcal{V}} && \big[\ \text{as } u(t) = \mathsf{ren}^{-1}\big(\xi(t)\big)\ \big]
\end{aligned}$$

Therefore, since (by assumption) f is surjective and $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$, our claim follows from Theorem 5. □

3.3 The View Complement

We have seen how injectivity of views ensures that, for each translatable view update, there is only one possible and acceptable way of consistently propagating the changes towards the base relations by means of a suitable database update. Since an injective view is lossless, in such a case the update is essentially performed on a restructured copy of the whole database, in the sense that the full informative content of the database is available also in the view, though by means of a different schema. It is easy to imagine situations in which this is not case, and sometimes even undesirable. For instance, consider the most common scenario in which user views are created in order to allow access to specific portions of the database, keeping untouched the rest of the data that is beyond the scope of the view. Since such views are lossy by design, the results achieved so far are not directly applicable.

We have seen before, when Lemma 1 was introduced, that translatable view updates may have unique translations even when the view is not injective. However, as already noted, we do not pursue such a direction here. Rather, given a view that is not injective, we want to gain injectivity by means of some additional information from the database, so as to fall back in the case discussed in the previous section.

The lack of injectivity in a view causes a loss of information, due to the fact that distinct database states are mapped to the same view state. Then, in order to distinguish between distinct database states, we need some extra "hints" that, combined with what is already known from the view itself, give a full account of the database content. This additional information is made available by another view, which takes the name of *view complement*, because it "complements" the partial information of a lossy view.

For the rest of this section, let $\mathcal{R}$, $\mathcal{V}$ and $\mathcal{W}$ be pairwise disjoint signatures, and let Σ and Γ be finite satisfiable sets of constraints over $\mathcal{R} \cup \mathcal{V}$ and $\mathcal{R} \cup \mathcal{W}$, respectively, such that their union is consistent.

Definition 6 (View complement). Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ and let g be a view from $\mathcal{R}$ to $\mathcal{W}$ under Γ. We say that g is a *complement* of f iff the following hold:

(1) $\mathbf{S}_\Sigma = \mathbf{S}_\Gamma$; and

(2) $\forall s, s' \in \mathbf{S}_\Sigma,\ s \neq s' \wedge f(s) = f(s') \implies g(s) \neq g(s')$.

In addition, g is said to be *tight* iff it also holds that:

(3) $\forall s, s' \in \mathbf{S}_\Sigma,\ g(s) \neq g(s') \implies f(s) = f(s')$. ■

In other words, 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. Moreover, a view complement g is tight when it separates only distinct database states between which f is not able to distinguish, that is, g complements f only where it is strictly necessary. Note that there exists at least one complement for every view, namely the "identity" mapping over the whole database.

The idea of view complement was first introduced by Bancilhon and Spyratos in [2]. Our definition is indeed based on their work (cf. Theorem 4.2 in [2]) with the additional requirement that f and g must have the same domain, which has to be explicitly enforced here as we are in a setting with views under constraints.

Note that the notion of view complement, tight or not, is symmetric, in the sense that if g is a (tight) complement of f then f is a (tight) complement of g. Indeed, the second condition of Definition 6 is equivalent to

$$\forall s, s' \in \mathbf{S}_\Gamma,\ s \neq s' \wedge g(s) = g(s') \implies f(s) \neq f(s') \ ;$$

while the last condition is equivalent to

$$\forall s, s' \in \mathbf{S}_\Gamma,\ f(s) \neq f(s') \implies g(s) = g(s') \ .$$

Therefore, we sometimes simply say that two views f and g are "complementary".

3.4 Injectivity by Complement

A view complement provides information that, added to the data coming from the view alone, allows to reconstruct the entire database. In what follows, again by means of logical definability, we will constructively characterise the notion of view complement and show how it is possible to obtain an injective view by opportunely combining a (lossy) view with its complement.

In light of Lemma 5, 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 $f \uplus g$.

Corollary 7. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ and g be a view from $\mathcal{R}$ to $\mathcal{W}$ under Γ. Then, the function $f \uplus g$ associating each $s \in \boldsymbol{S}_\Sigma \cap \boldsymbol{S}_\Gamma$ with the state $f(s) \uplus g(s)$ is a view from $\mathcal{R}$ to $\mathcal{V} \cup \mathcal{W}$ under $\Sigma \cup \Gamma$.*

Proof. For each $s \in \mathbf{S}_\Sigma \cap \mathbf{S}_\Gamma$, we have that $s \uplus f(s) \models \Sigma$, as f is a view under Σ, and $s \uplus g(s) \models \Gamma$, as g is a view under Γ. Therefore, $s \uplus f(s) \uplus g(s) \models \Sigma \cup \Gamma$ by Lemma 5. □

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.

LEMMA 6. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let g be a view from $\mathcal{R}$ to $\mathcal{W}$ under Γ and let $\boldsymbol{S}_\Sigma = \boldsymbol{S}_\Gamma$. Then, $f \uplus g$ is injective if and only if f and g are complementary.*

Proof. We will show that f and g are not complementary if and only if $f \uplus g$ is not injective. Assuming $\mathbf{S}_\Sigma = \mathbf{S}_\Gamma$, we have that f and g are not complementary iff the second condition of Definition 6 is violated, that is, iff there exist states $s, s' \in \mathbf{S}_\Sigma$ with $s \neq s'$ and such that $f(s) = f(s')$ and $g(s) = g(s')$. Since f and g are views under constraints, $f(s)$ and $g(s)$ are states with the same domain over disjoint signatures, and so are $f(s')$ and $g(s')$. Therefore, we have that $f(s) = f(s')$ and $g(s) = g(s')$ iff $f(s) \uplus g(s) = f(s') \uplus g(s')$, that is, iff $f \uplus g$ is not injective. □

The following lemma establishes that, given views f and g induced by Σ and Γ, respectively, the set $\Sigma \cup \Gamma$ in turn induces a view that is the union of f and g. Moreover, the views have the same domain if and only if the embeddings of their corresponding constraints are equivalent.

LEMMA 7. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$ and $\mathcal{R} \twoheadrightarrow^g_\Gamma \mathcal{W}$. Denote by $\widetilde{\Sigma}_\mathcal{R}$ and $\widetilde{\Gamma}_\mathcal{R}$ the $\mathcal{R}$-embeddings of Σ and Γ, respectively. Then,*

1) $\mathcal{R} \xrightarrow[\Sigma\cup\Gamma]{f\uplus g} \mathcal{V} \cup \mathcal{W}$;
2) $\boldsymbol{S}_\Sigma = \boldsymbol{S}_\Gamma$ if and only if $\widetilde{\Sigma}_\mathcal{R} \equiv \widetilde{\Gamma}_\mathcal{R}$.

Proof.

1) Assume $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ and $\mathcal{R} \twoheadrightarrow_\Gamma \mathcal{W}$, and suppose it does not hold that $\mathcal{R} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{V} \cup \mathcal{W}$. Hence, there are two models $s \uplus t \uplus z$ and $s \uplus t' \uplus z'$ of $\Sigma \cup \Gamma$ such that $t \neq t'$ or $z \neq z'$. When $t \neq t'$, as both $s \uplus t$ and $s \uplus t'$ are models of Σ by Lemma 5, we get a contradiction of $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$. Similarly, $s \uplus z$ and $s \uplus z'$ are models of Γ; thus, if $z \neq z'$, we get a contradiction of $\mathcal{R} \twoheadrightarrow_\Gamma \mathcal{W}$. Therefore, $\mathcal{R} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{V} \cup \mathcal{W}$. Now, let f be the view from $\mathcal{R}$ to $\mathcal{V}$ induced by Σ and g be the view from $\mathcal{R}$ to $\mathcal{W}$ induced by Γ. We are left to show that $f \uplus g$ is the view from $\mathcal{R}$ to $\mathcal{V} \cup \mathcal{W}$ induced by $\Sigma \cup \Gamma$, which follows by Theorem 1 from Corollary 7 and the fact that $\mathcal{R} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{V} \cup \mathcal{W}$.
2) Direct consequence of Corollary 4, for which $s \in \mathbf{S}$ is Σ-consistent iff $s \models \widetilde{\Sigma}_\mathcal{R}$ and it is Γ-consistent iff $s \models \widetilde{\Gamma}_\mathcal{R}$. □

PROPOSITION 2. *Let $\mathcal{R} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{V} \cup \mathcal{W}$ and $\boldsymbol{S}_\Sigma = \boldsymbol{S}_\Gamma$. Then, $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ and $\mathcal{R} \twoheadrightarrow_\Gamma \mathcal{W}$.*

Proof. Suppose $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ does not hold, that is, there exist models $s \uplus t$ and $s \uplus t'$ of Σ with $t \neq t'$, for some $s \in \mathbf{S}_\Sigma$. As $\mathbf{S}_\Sigma = \mathbf{S}_\Gamma$, we have that $s \in \mathbf{S}_\Gamma$, hence there exists a state z over $\mathcal{W}$ for which $s \uplus z \models \Gamma$. But then, by Lemma 5, $s \uplus t \uplus z$ and $s \uplus t' \uplus z$ are distinct models of $\Sigma \cup \Gamma$, in contradiction of $\mathcal{R} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{V} \cup \mathcal{W}$. A symmetric argument applies for proving $\mathcal{R} \twoheadrightarrow_\Gamma \mathcal{W}$. □

We are now ready to give an important characterisation of complementarity between two views induced by constraints in terms of logical equivalence and definability.

THEOREM 8. *Let* $\mathcal{R} \twoheadrightarrow^{f}_{\Sigma} \mathcal{V}$ *and* $\mathcal{R} \twoheadrightarrow^{g}_{\Gamma} \mathcal{W}$. *Then,* f *and* g *are complementary iff* $\widetilde{\Sigma}_{\mathcal{R}} \equiv \widetilde{\Gamma}_{\mathcal{R}}$ *and* $\mathcal{V} \cup \mathcal{W} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{R}$.[3]

Proof. By Lemma 7, $\Sigma \cup \Gamma$ induce a view from $\mathcal{R}$ to $\mathcal{V} \cup \mathcal{W}$ that is exactly $f \uplus g$ and, moreover, we have that $\mathbf{S}_\Sigma = \mathbf{S}_\Gamma$ iff $\widetilde{\Sigma}_{\mathcal{R}} \equiv \widetilde{\Gamma}_{\mathcal{R}}$. Then, by Lemma 6, f and g are complementary iff $f \uplus g$ is injective, which by Lemma 3 is the case if and only if $\mathcal{V} \cup \mathcal{W} \twoheadrightarrow_{\Sigma\cup\Gamma} \mathcal{R}$. □

Given a view f induced by a set of constraints Σ, one way of finding a complement consists in abducing another set of constraints Γ consistent with Σ and satisfying the conditions of Theorem 8, which guarantees that the view g induced by such a Γ is indeed a complement of f.

In the general case considered by B&S, where views are arbitrary functions, there are views not induced by constraints that could nevertheless qualify as complements of the given view. However, without a constructive characterisation for such views, like the one we propose here for views induced by constraints, there are no means to find and compute them.

3.5 Translation under Constant Complement

We have seen that, by means of a view complement, we can recover information missing from a lossy view. In particular, we have shown that the the union of a view with its complement is an injective view, which gives us the possibility of using the results presented in Section 3.2. However, as pointed out earlier, updating an injective view, due to its losslessness, is essentially the same as updating the whole database through a different schema. Considering that lossy views are meant to limit the interaction with the database in the first place, such an unrestricted update capability is undesirable.

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. Putting together translatability of updates and invariance of the complement results in the formal notion given below (cf. Definition 5.1 in [2]).

DEFINITION 7 (g-translatability). Let f be a view under Σ and let g be a complement of f. A view update $u \in U_{\mathcal{V}}$ is called *g-translatable* iff for each $s \in \mathbf{S}_\Sigma$ there exists $s' \in \mathbf{S}_\Sigma$ such that:

(1) $f(s') = uf(s)$; and (translatability)
(2) $g(s') = g(s)$. (constant complement) ■

That is, a view update is g-translatable if it is translatable (according to Definition 5) and, in addition, leaves the complement g unchanged. For this reason,

[3] $\widetilde{\Sigma}_{\mathcal{R}}$ and $\widetilde{\Gamma}_{\mathcal{R}}$ denote the $\mathcal{R}$-embeddings of Σ and Γ, respectively.

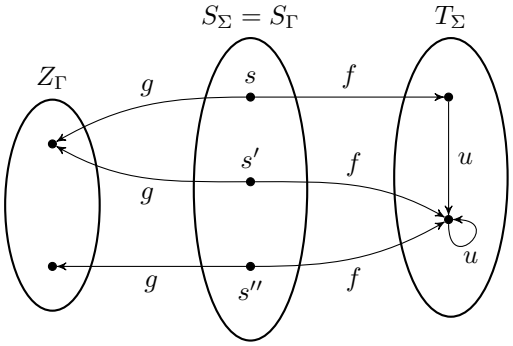

Figure 7: The complement g is not tight, as $f(s) \neq f(s'')$ even though $g(s) \neq g(s'')$. Nevertheless, the update u is g-translatable.

we say that such an update is *translatable under constant complement*. 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. Thus, the choice of a complement defines an "update policy", assigning unambiguous semantics to the view updates. It is easy to see that non-strict updates are always g-translatable, independently on g itself. On the other hand, strict updates are translatable under constant complement only if the complement is not injective.

PROPOSITION 3. *Let f be a view under Σ, let g be a complement of f and let $u \in U_f$. Then, u is g-translatable only if g is not injective.*

Proof. Since u is strict, $f(s) \neq uf(s)$ for some state $s \in \mathbf{S}_\Sigma$. Assuming u to be g-translatable, there is $s' \in \mathbf{S}_\Sigma$ such that $f(s') = uf(s)$ and $g(s') = g(s)$. Hence, we get $f(s') \neq f(s)$ and, in turn, $s' \neq s$. Therefore, g is not injective. □

Intuitively, a strict update is bound to make some changes on the view, which are then propagated to the database, which is in turn forced to change, and an injective complement is nothing else than a lossless view of the database, therefore the changes must be reflected in it as well.

Next, we show that tightness is a sufficient condition for the complement to remain constant with respect to a translatable update.

PROPOSITION 4. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $u \in U_\mathcal{V}$ be translatable and let g be a tight complement of f. Then, u is g-translatable.*

Proof. Let $s \in \mathbf{S}_\Sigma$. We need to show that there is a database state whose image through f is equal to $uf(s)$ and whose image through g is equal to $g(s)$. Clearly, if $uf(s) = f(s)$, the state we are looking for is s itself. Thus, let us consider the non-trivial case when $uf(s) \neq f(s)$. As u is translatable, $uf(s)$ is in the image of f, hence there exists $s' \in \mathbf{S}_\Sigma$ such that $f(s') = uf(s)$. Since $f(s') \neq f(s)$, by the tightness of g we obtain that $g(s') = g(s)$. □

Not surprisingly, the converse of Proposition 4 does not hold, in that translatability under constant complement does not imply the tightness of the complement, as the counter-example in Figure 7 demonstrates.

Given two updates $u \in U_{\mathcal{V}}$ and $v \in U_{\mathcal{W}}$, with some abuse of notation we denote by $u \uplus v$ the combined update that, for states t and z with the same domain over $\mathcal{V}$ and $\mathcal{W}$, respectively, associates $t \uplus z$ with $u(t) \uplus v(z)$. 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 9. *Let f and g be complementary, let $u \in U_{\mathcal{V}}$ and let $v \in U_{\mathcal{W}} \setminus U_g$. Then, u is g-translatable w.r.t. f if and only if $u \uplus v$ is translatable w.r.t. $f \uplus g$.*

Proof. The update $u \uplus v$ is translatable w.r.t. $f \uplus g$ iff, for each $s \in \mathbf{S}_\Sigma$, there is $s' \in \mathbf{S}_\Sigma$ s.t. $\big(f \uplus g\big)(s') = \big(u \uplus v\big)\big((f \uplus g)(s)\big)$. By definition of $f \uplus g$ and $u \uplus v$, this is the case iff $f(s') \uplus g(s') = uf(s) \uplus vg(s)$ and, in turn, iff

$$f(s') = uf(s) \text{ and } g(s') = vg(s) \ . \tag{14}$$

Since $v \notin U_g$ (that is, v is non-strict on g), v is the identity on $g(\mathbf{S}_\Sigma)$, hence $vg(s) = g(s)$ for every $s \in \mathbf{S}_\Sigma$. Therefore, we have $g(s') = g(s)$ in (14), which consequently becomes the definition of g-translatability of u w.r.t. f. □

This allows us to extend the result on the translatability of updates on injective views, obtained in the previous section, also to the case of g-translatability.

THEOREM 10. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$, let $\mathcal{R} \twoheadrightarrow^g_\Gamma \mathcal{W}$ and let g be a complement of f. Let $\widetilde{\Pi}$ be the $(\mathcal{V} \cup \mathcal{W})$-embedding of $\Sigma \cup \Gamma$ and let ren be a renaming over $\mathcal{R} \cup \mathcal{V} \cup \mathcal{W}$. Let $u \in U_{\mathcal{V}}$ be expressed by Ξ and let Ω be the $\mathcal{W}$-defining set such that $\forall \overline{x} \,.\, W(\overline{x}) \equiv \mathsf{ren}\big(W(\overline{x})\big)$ for each $W \in \mathcal{W}$. Then, u is g-translatable if and only if $\widetilde{\Pi} \cup \Xi \cup \Omega \models \mathsf{ren}(\widetilde{\Pi})$.*

Proof. Let ω denote the function induced by Ω. Then, $z = \mathsf{ren}^{-1}\big(\omega(z)\big)$ for every $z \in g(\mathbf{S}_\Sigma)$, hence Ω expresses an update $v \in U_{\mathcal{W}} \setminus U_g$. By Theorem 9, u is g-translatable w.r.t. f if and only if the update $u \uplus v$ expressed by $\Xi \cup \Omega$ is translatable w.r.t. $f \uplus g$ and, by Theorem 6, this is the case iff $\widetilde{\Pi} \cup \Xi \cup \Omega \models \mathsf{ren}\big(\widetilde{\Pi}\big)$. □

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 [7] where only two views are considered. Indeed, our general criterion for translatability of updates w.r.t. an instance under egd's and full tgd's subsumes the conditions given in [7] 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 following open formula:

$$\mathsf{project}_X(\overline{x}) \stackrel{\text{def}}{\equiv} \exists \overline{y}\, R(\overline{w}) \ ; \tag{15}$$

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 one formula of the following form:

$$\forall \overline{x} \left(V_i(\overline{x}) \leftrightarrow \mathsf{project}_{X_i}(\overline{x}) \right) \ ; \tag{16}$$

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 $\{1, \ldots, \mathsf{arity}(R)\}$ 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]).

4.1 Invertibility of views

Let us first consider the case, studied in [7], in which there are only two view symbols, that is, $\mathcal{V} = \{V_1, V_2\}$. Rephrasing the definition given in [7], 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 $R^s = R^{s'}$. Observe 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 $\mathsf{impl\text{-}def}_{\text{fin}}(R, \mathcal{V}, \Sigma)$, where $\mathsf{impl\text{-}def}_{\text{fin}}$ denotes $\mathsf{impl\text{-}def}$ restricted to finite models (i.e., using $\models_{\text{fin}}$ in place of $\models$). It is shown in [7] 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 $\mathsf{impl\text{-}def}_{\text{fin}}(R, \mathcal{V}, \Sigma)$ 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 $\mathcal{V} = \{V_1, \ldots, V_n\}$ 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 [7], 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 following equivalence:

$$\forall \overline{x} \left(R(\overline{x}) \leftrightarrow V_1(\overline{x}_1) \wedge \cdots \wedge V_n(\overline{x}_n) \right) \ ; \tag{17}$$

where $\overline{x}$ and $\overline{x}_1, \ldots, \overline{x}_n$ are sequences of variables such that:

1. $\mathsf{var}(\overline{x}) = \bigcup_{i=1}^{n} \mathsf{var}(\overline{x}_i)$,
2. $\overline{x}_i[p] = \overline{x}_j[q]$ iff $\mathsf{pos}(V_i, p) = \mathsf{pos}(V_j, q)$, and
3. $\overline{x}[p] = \overline{x}_i[q]$ iff $p = \mathsf{pos}(V_i, q)$.

In other words, variables corresponding to the same position in R must coincide. Note that (17) is well-defined if and only if $\{1, \ldots, \mathsf{arity}(R)\} = \bigcup_{i=1}^{n} \mathsf{pos}(V_i)$, that is, the projections cover the positions of R entirely.

4.2 Translatability of view updates

We now turn to the problem 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 $u(t)$,[4]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 $V_1(\overline{x}_1) \wedge \cdots \wedge V_n(\overline{x}_n)$.

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 [7], 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 [7] 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.[5] We give an idea of how our criterion corresponds to the conditions of [7] in the case of insertions by means of an example.

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

[5]For instance, condition (b) of Theorem 3 in [7] is necessary only due to the assumption that the tuple to be inserted is not already present in the view extension.

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 such that $E < D < P < S < M$, hence $\mathsf{apos}(E) = 1$, $\mathsf{apos}(D) = 2$, $\mathsf{apos}(P) = 3$, $\mathsf{apos}(S) = 4$ and $\mathsf{apos}(M) = 5$. Let $\mathcal{V} = \{V_1, V_2\}$ and $\Sigma_{\mathcal{RV}}$ consist of:

$$\forall x_1, x_2, x_3 \quad \big(V_1(x_1, x_2, x_3) \quad \leftrightarrow \exists y_1, y_2\, R(x_1, x_2, x_3, y_1, y_2)\big)\ ; \tag{18a}$$

$$\forall x_1, x_2, x_3, x_4\, \big(V_2(x_1, x_2, x_3, x_4) \leftrightarrow \exists y_1 \quad R(x_1, x_2, y_1, x_3, x_4)\big)\ ; \tag{18b}$$

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

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \wedge R(x_1, x_2, x_3', x_4', x_5') \rightarrow x_3 = x_3'\big)\ ; \tag{19a}$$

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \wedge R(x_1', x_2', x_3, x_4', x_5') \rightarrow x_4 = x_4'\big)\ ; \tag{19b}$$

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \wedge R(x_1', x_2, x_3', x_4', x_5') \rightarrow x_5 = x_5'\big)\ ; \tag{19c}$$

where $\overline{x}$ is the sequence of all the variables appearing in each case. Equations (19a), (19b) and (19c) express the fd's $ED \rightarrow P$, $P \rightarrow S$ and $D \rightarrow M$, respectively. It can be verified that $\Sigma = \Sigma_{\mathcal{R}} \cup \Sigma_{\mathcal{RV}}$ implies the jd $\bowtie[EDP, EDSM]$, that is:

$$\forall \overline{x}\, \big(R(x_1, x_2, x_3, x_4, x_5) \leftrightarrow V_1(x_1, x_2, x_3) \wedge V_2(x_1, x_2, x_4, x_5)\big)\ . \tag{20}$$

By substituting every occurrence of $R(x_1, x_2, x_3, x_4)$ in (18a), (18b), (19a), (19b) and (19c) with the explicit definition $V_1(x_1, x_2, x_3) \wedge V_2(x_1, x_2, x_4, x_5)$ we get the tgd's:

$$\forall x_1, x_2, x_3 \quad \big(V_1(x_1, x_2, x_3) \quad \rightarrow \exists y_1, y_2\, V_2(x_1, x_2, y_1, y_2)\big)\ ; \tag{21a}$$

$$\forall x_1, x_2, x_3, x_4\, \big(V_2(x_1, x_2, x_3, x_4) \rightarrow \exists y_1 \quad V_1(x_1, x_2, y_1) \quad \big)\ ; \tag{21b}$$

and egd's:

$$\forall \overline{x}\, \big(V_1(x_1, x_2, x_3) \wedge V_1(x_1, x_2, x_3') \quad \rightarrow x_3 = x_3'\big)\ ; \tag{21c}$$

$$\forall \overline{x}\, \big(V_1(x_1, x_2, x_3) \wedge V_1(x_1', x_2', x_3) \wedge V_2(x_1, x_2, x_4, x_5) \wedge V_2(x_1', x_2', x_4', x_5') \rightarrow x_4 = x_4'\big)\ ; \tag{21d}$$

$$\forall \overline{x}\, \big(V_2(x_1, x_2, x_3, x_4) \wedge V_2(x_1', x_2, x_3', x_4') \rightarrow x_4 = x_4'\big)\ ; \tag{21e}$$

together constituting the $\mathcal{V}$-embedding $\widetilde{\Sigma}_{\mathcal{V}}$ of Σ. Note that, while the fd's (19a) and (19c) on R are preserved as the fd's (21c) and (21e) on V_1 and V_2, respectively, the fd (19b) becomes a genuine egd, namely (21d), 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 (21a), but it satisfies (21b) 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 [7] 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 the edg's in $\widetilde{\Sigma}_{\mathcal{V}}$, namely (21c), (21d) and (21e), corresponds to condition (c). ■

It should appear clear that with $\left|2^{\mathcal{V}}\right|$ choices for the complement symbols, stating conditions à la [7] 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 [7] 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 [7]. 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:

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

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 [7] 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 [7] 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 [6] in this sense;
2. in particular, further extend the setting presented in Section 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 [10, 5].

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Transactions on Database Systems*, 6(4):557–575, Dec. 1981.

[3] C. Beeri and M. Y. Vardi. On acyclic database decompositions. *Information and Control*, 61(2):75–84, 1984.

[4] E. W. Beth. On Padoa's method in the theory of definition. *Indagationes Mathematicae*, 15:330–339, 1953.

[5] A. Borgida, J. de Bruijn, E. Franconi, I. Seylan, U. Straccia, D. Toman, and G. Weddell. On finding query rewritings under expressive constraints. In *Proceedings of SEBD-2010*, Rimini, Italy, June 2010.

[6] A. Calì, G. Gottlob, and T. Lukasiewicz. Datalog$^{\pm}$: a unified approach to ontologies and integrity constraints. In *Proceedings of the 12th International*

Conference on Database Theory, ICDT '09, pages 14–30, New York, NY, USA, 2009. ACM.

[7] S. S. Cosmadakis and C. H. Papadimitriou. Updates of relational views. *Journal of the Association for Computing Machinery*, 31(4):742–760, Oct. 1984.

[8] G. Gottlob, P. Paolini, and R. Zicari. Properties and update semantics of consistent views. *ACM Transactions on Database Systems*, 13(4):486–524, Dec. 1988.

[9] Y. Gurevich. Toward logic tailored for computational complexity. In *Computation and Proof Theory*, volume 1104 of *Lecture Notes in Mathematics*, pages 175–216. Springer Berlin / Heidelberg, 1984.

[10] G. Huang. Constructing Craig interpolation formulas. In *Computing and Combinatorics*, volume 959 of *Lecture Notes in Computer Science*, pages 181–190. Springer Berlin / Heidelberg, 1995.

[11] J. Lechtenbörger. The impact of the constant complement approach towards view updating. In *Proceedings of PODS 2003*, pages 49–55, San Diego, CA, June 2003.