Symposium on Advances in Databases and Information Systems (ADBIS'98),
September 8-11, 1998, Poznan, Poland, LNCS, Springer-Verlag, Berlin, 1998.

Deriving Relationships between Integrity Constraints for Schema Comparison

Can Türker and Gunter Saake

Otto-von-Guericke-Universität Magdeburg
Institut für Technische und Betriebliche Informationssysteme
Postfach 4120, D–39016 Magdeburg, Germany
{tuerker|saake}@iti.cs.uni-magdeburg.de

Abstract. Schema comparison is essential for integrating different database schemata. Since the semantics of a schema is also represented by its integrity constraints, they must be considered by a correct schema comparison method. Especially the extensional relationships between classes are determined by the relationship between the corresponding integrity constraint sets. In this paper, we work out the relationships between different types of integrity constraints. As a result, we present rules for comparing integrity constraint sets. These rules can be used after a schema conforming step, where naming, type and structural conflicts are solved, to exactly fix the extensional relationship between object classes.

1 Introduction

Schema integration [1] is one of the central issues in logically integrating database systems [6]. Schema integration aims at deriving an integrated schema which is a virtual view on all database classes to be integrated. Traditionally, all existing schema integration methods, e.g. [4, 7, 3, 2, 5], assume that the database designer – as integrator – is an expert of the application domain who has complete knowledge about the semantics of the schemata to be integrated. Thus, the database designer should be able to exactly define the extensional relationship between two classes of different schemata in the schema comparison step. Obviously, this is a very hard task for the database designer.

A key point which is not regarded by the existing integration approaches is the correspondence between integrity constraints and extensional relationships. In case a schema captures all relevant integrity constraints occurring in the modeled real world, the extensional relationships between the classes are completely determined by the corresponding integrity constraints defined on these classes. Thus, our idea is to exploit the relationship between the integrity constraints of two classes to derive the correct extensional relationship between these classes.

In this paper, we work out the problem of relating different sets of integrity constraints. This issue is essential for comparing two classes of different schemata correctly. However, the comparison of arbitrary constraints is undecidable. Hence, we restrict ourselves to a certain set of constraint types. For these types we provide rules for computing the relationship between two integrity constraints. These rules can be used to extend existing schema comparison methods.

The paper is organized as follows: Section 2 investigates rules for comparing integrity constraint sets. The application of these rules is presented in Section 3. Finally, a short outlook on future work concludes the paper.

2 Relationships between Integrity Constraint Sets

Definition 1. A *set of object integrity constraints* $\mathcal{IC}$ restricts the set of possible objects, i.e. the database universe $\mathcal{U}$, to the semantically correct objects $\mathcal{U}^{\mathcal{IC}}$. In the sequel, we use the term *integrity constraint set* to refer to such a set. From a logical point of view, an integrity constraint set corresponds to a conjunction of integrity constraints. An integrity constraint itself is a formula. □

Objects with similar properties can be grouped into *classes*, i.e., classes are subsets of the database universe. The integrity constraint set of a class C, denoted as $\mathcal{IC}_C$, defines the semantically correct objects of that class. Additionally, we also consider uniqueness constraints which describes correct sets of database states.

Definition 2. Two constraints IC_1 and IC_2 defined on the same variables (attributes) are called *directly related* by one the following relationships:

$$\begin{aligned}
IC_1 \varnothing IC_2 &:\Leftrightarrow \mathcal{U}^{IC_1} \cap \mathcal{U}^{IC_2} = \varnothing \\
IC_1 \equiv IC_2 &:\Leftrightarrow \mathcal{U}^{IC_1} \equiv \mathcal{U}^{IC_2} \\
IC_1 \supset IC_2 &:\Leftrightarrow \mathcal{U}^{IC_1} \subset \mathcal{U}^{IC_2} \\
IC_1 \Cap IC_2 &:\Leftrightarrow \neg((IC_1 \varnothing IC_2) \vee (IC_1 \equiv IC_2) \vee (IC_1 \subset IC_2) \vee (IC_1 \supset IC_2))
\end{aligned}$$

Two constraints defined on different variables can be *transitively* related if there is another constraint defined on both variables. Two constraints are called *unrelated* ($IC_1 \Cup IC_2$) if they are neither directly nor transitively related. □

In order to relate integrity constraint sets to each other, *implicit constraints* must be considered. The following example illustrates the application of implicit constraints. Let there be two integrity constraint sets given as follows:

$$\begin{aligned}
\mathcal{IC}_1 &= \{(\mathsf{x} < \mathsf{y}), (\mathsf{y} < \mathsf{z})\} \\
\mathcal{IC}_2 &= \{(\mathsf{x} < \mathsf{z})\}
\end{aligned}$$

Comparing each pair of constraints of the given sets, results that these sets must be unrelated. However, since the conjunction of $(\mathsf{x} < \mathsf{y})$ and $(\mathsf{y} < \mathsf{z})$ implies $(\mathsf{x} < \mathsf{z})$, we can derive that $\mathcal{IC}_1 \supset \mathcal{IC}_2$ holds. For our following considerations we use the notion of *extended integrity constraint set* to denote an integrity constraint set capturing all its implicit constraints. Since the implication problem is not decidable for general constraints, we restrict our considerations to the following types of constraints[1]:

C1: $\mathsf{x}_1\, \theta\, \mathsf{c}_1$

[1] We are aware of that there are other types of constraints which also decidable. However, this issue is not focused in this paper.

C2: $\mathsf{x}_1\,\theta\,\mathsf{x}_2$

C3: **unique**$(\mathsf{x}_1, ..., \mathsf{x}_n)$

C4: $\mathsf{C}_i \Rightarrow \mathsf{C}_j$, where C_i and C_j are constraints of type C1 or C2

The symbol $\theta \in \{<, \leq, =, \neq, \geq, >\}$ stands for a comparison operator, x_1, ..., x_n are variables (attributes), and c_1 is a constant.

Relationships between C1*-Constraints.* A C1-constraint restricts the possible values of exactly one attribute. Obviously, when we compare two C1-constraints, the resulting relationship between such two constraints depends on the respective comparison operators and constants:

$$\underbrace{(\mathsf{x}_1\,\theta_1\,\mathsf{c}_1)}_{IC_1} \wedge \underbrace{(\mathsf{x}_2\,\theta_2\,\mathsf{c}_2)}_{IC_2} \wedge (\mathsf{c}_1\theta_3\mathsf{c}_2) \Rightarrow (IC_1\,\Theta_4\,IC_2)$$

The comparison operators θ_1 and θ_2 are in $\{<, \leq, =, \neq, \geq, >\}$, whereas the constant relationship operator θ_3 is in $\{<, =, >\}$. Thus, there are 108 $(6 \times 6 \times 3)$ possible combinations (for the antecedent of the implication). Fig. 1 summarizes the rules for computing the relationship Θ_4 $(\in \{\varnothing, \supset, \equiv, \subset, \Cap\})$ between two given C1-constraints. The symbol ϑ stands for one of the comparison operators $\{<, \leq, =, \neq, \geq, >\}$. The tenth row states that two C1-constraints are equivalent if and only if they have the same comparison operators and constants. In Fig. 1,

$(\mathsf{x}_1\ \theta_1\ \mathsf{c}_1) \wedge (\mathsf{x}_2\ \theta_2\ \mathsf{c}_2) \wedge (\mathsf{c}_1\ \theta_3\ \mathsf{c}_2) \Rightarrow (IC_1\ \Theta_4\ IC_2)$			
θ_1	θ_2	θ_3	Θ_4
$<, \leq, =$	$=, \geq, >$	$<$	$\varnothing$
$<$	$=, \geq, >$	$=$	$\varnothing$
$\leq$	$>$	$=$	$\varnothing$
$=$	$<, \neq, >$	$=$	$\varnothing$
$=$	$<, \leq, =$	$>$	$\varnothing$
$<, \leq, =$	$<, \leq, \neq$	$<$	$\supset$
$<$	$\leq, \neq$	$=$	$\supset$
$=$	$\leq, \geq$	$=$	$\supset$
$=$	$\neq, \geq, >$	$>$	$\supset$
ϑ	ϑ	$=$	$\equiv$
$<, \leq$	$<, \leq, =$	$>$	$\subset$
$\leq$	$<, =$	$=$	$\subset$
$<, \leq$	$\neq, \geq, >$	$>$	$\Cap$
$\leq$	$\neq, \geq$	$=$	$\Cap$

Fig. 1. Rules for Computing Relationships between C1-Constraints

we have only listed the rules for the combinations where at least one comparison operator is in $\{<, \leq, =\}$. Based on these rules we can also derive relationships between constraints basing on the "complement" comparison operators $\{\geq, >, \neq\}$.

For that, we first have to build the complement constraints, i.e. change the comparison operators by their complements ($<$ for $\geq$, $\leq$ for $>$, and $=$ for $\neq$). Then, we compute the relationship for the complement constraints according to the rules depicted in Fig. 1. Finally, the result is "reversed" using the following complement rules for relationships: $(\varnothing \leftrightsquigarrow \Cap), (\supset \leftrightsquigarrow \subset), (\equiv \leftrightsquigarrow \equiv)$.

Relationships between C2-*Constraints.* Similar to C1-constraints, rules can be derived for computing relationships between C2-constraints. Two C2-constraints are directly related if they are defined on the same variables. The relationship type is determined by the corresponding comparison operators, e.g. the constraint $(\mathsf{x} > \mathsf{y})$ implies the constraint $(\mathsf{x} \geq \mathsf{y})$:

$$\underbrace{(\mathsf{x}\,\theta_1\,\mathsf{y})}_{IC_1} \land \underbrace{(\mathsf{x}\,\theta_2\,\mathsf{y})}_{IC_2} \Rightarrow (IC_1\;\Theta_3\;IC_2)$$

The comparison operators θ_1 and θ_2 are in $\{<, \leq, =, \neq, \geq, >\}$. Fig. 2(a) summarizes the rules for computing Θ_3 $(\in \{\varnothing, \supset, \equiv, \subset, \Cap\})$ between C2-constraints. The symbol ϑ stands for one of possible comparison operators. Analogously to Fig. 1, Fig. 2(a) contains only the rules for the operators $\{<, \leq, =\}$. Fig. 2(b) presents rules to derive an implicit C2-constraint from two C2-constraints:

$$(\mathsf{x}\,\theta_1\,\mathsf{y}) \land (\mathsf{y}\,\theta_2\,\mathsf{z}) \Rightarrow (\mathsf{x}\,\theta_3\,\mathsf{z})$$

The comparison operators θ_1, θ_2, and θ_3 are in $\{<, \leq, =, \neq, \geq, >\}$, whereas x, y and z are variables (attributes). Please note that these rules also hold when z is a constant. In this case, we derive an implicit C1-constraint from a pair of (C2, C1)-constraints. Example 1 illustrates this fact.

$(x\ \theta_1\ y) \land (x\ \theta_2\ y) \Rightarrow (IC_1\ \Theta_3\ IC_2)$		
θ_1	θ_2	Θ_3
$<$	$=, \geq, >$	$\varnothing$
$\leq$	$>$	$\varnothing$
$=$	$<, \neq, >$	$\varnothing$
$<$	$\leq, \neq$	$\supset$
$=$	$\leq, \geq$	$\supset$
ϑ	ϑ	$\equiv$
$\leq$	$<, =$	$\subset$
$\leq$	$\neq, \geq$	$\Cap$

$(\mathsf{x}\ \theta_1\ \mathsf{y}) \land (\mathsf{y}\ \theta_2\ \mathsf{z}) \Rightarrow (\mathsf{x}\ \theta_3\ \mathsf{z})$		
θ_1	θ_2	θ_3
$<$	$<, \leq$	$<$
$<, \leq$	$<$	$<$
$\leq$	$\leq$	$\leq$
$=$	ϑ	ϑ
ϑ	$=$	ϑ
$\geq$	$\geq$	$\geq$
$>$	$\geq, >$	$>$
$\geq, >$	$>$	$>$

Fig. 2. Rules for (a) C2-Constraints and (b) Implicit C2-Constraints

Example 1. Suppose that the following integrity constraint sets are given:

$$\mathcal{IC}_1 = \{(\mathsf{x} \leq \mathsf{y})\}$$
$$\mathcal{IC}_2 = \{(\mathsf{x} \geq \mathsf{y})\}$$

According to the last rule in Fig. 2, $\mathcal{IC}_1$ and $\mathcal{IC}_2$ are overlapping. $\square$

Example 2. Assume that the following integrity constraint sets are given:

$$\mathcal{IC}_1 = \{(\mathsf{x} > \mathsf{y}), (\mathsf{y} \geq 4)\}$$
$$\mathcal{IC}_2 = \{(\mathsf{x} > 2)\}$$
$$\mathcal{IC}_3 = \{(\mathsf{x} \geq \mathsf{y}), (\mathsf{y} \geq \mathsf{z})\}$$
$$\mathcal{IC}_4 = \{(\mathsf{x} > \mathsf{z})\}$$

According to the last but one rule in Fig. 2(b), the constraints $(\mathsf{x} > \mathsf{y})$ and $(\mathsf{y} \geq 4)$ imply $(\mathsf{x} > 4)$. Since $(\mathsf{x} > 4)$ implies $(\mathsf{x} > 2)$, we can state that $\mathcal{IC}_1 \supset \mathcal{IC}_2$ holds (cf. Fig. 1). Using rule six in Fig. 2(b) we can derive $(\mathsf{x} \geq \mathsf{z})$ from the constraints $(\mathsf{x} \geq \mathsf{y})$ and $(\mathsf{y} \geq \mathsf{z})$. Since the constraint $(\mathsf{x} > \mathsf{z})$ implies the constraint $(\mathsf{x} \geq \mathsf{z})$, we may reason that $\mathcal{IC}_3 \Cap \mathcal{IC}_4$ holds (cf. Fig. 2(a)). □

Relationships between C3*-Constraints.* A uniqueness constraint (type C3) is defined on a set of attributes. The effect of a uniqueness constraint is that the values of the respective attribute combinations must be unique. The following theorem defines the relationships between two uniqueness constraints depending on the corresponding attribute sets.

Theorem 1 (Relationships between Uniqueness Constraints). Let x and y be two sets of attributes. Then the following rules hold:

$$(\textbf{unique}(\mathsf{x}) \equiv \textbf{unique}(\mathsf{y})) \text{ iff } (\mathsf{x} = \mathsf{y})$$
$$(\textbf{unique}(\mathsf{x}) \supset \textbf{unique}(\mathsf{y})) \text{ iff } (\mathsf{x} \subset \mathsf{y})$$

From (semantically) overlapping or disjoint attribute sets we can only reason that the respective uniqueness constraints overlap. □

Proof. A uniqueness constraint $\textbf{unique}(\mathsf{x}_1, ..., \mathsf{x}_n)$ can be expressed as follows:

$$(\forall \mathsf{o}_1, \mathsf{o}_2 \in U(\mathsf{o}_1.\mathsf{x}_1 = \mathsf{o}_2.\mathsf{x}_1 \wedge ... \wedge \mathsf{o}_1.\mathsf{x}_n = \mathsf{o}_2.\mathsf{x}_n \Rightarrow \mathsf{o}_1 = \mathsf{o}_2))$$

Such a formula is weakened by strengthen the antecedent of the implication. This can be done by considering further attributes, i.e., by adding further terms to the conjunction on the antecedent of the implication. In consequence, a uniqueness constraints is stronger than another uniqueness constraint if it is defined on a subset of the attributes of the other one. Considering this fact in both directions, we can derive that two uniqueness constraints are equivalent if and only if they are formulated over the same set of attributes.

In case the attribute sets are overlapping or disjoint, the antecedents of both implications contain attributes which occur only in one of the respective formulas. Thus, the antecedents are not comparable and we have to conclude that such kinds of uniqueness constraints are overlapping. □

Example 3. Let there be a class Employee with the attributes ssn, name, salary, and address. If a constraint **unique**(ssn) hold on class Employee, then the constraint **unique**(ssn, name, salary) must also hold on this class. In this case, the latter constraint is a specialization of the former one. On the other hand, the constraints **unique**(ssn) and **unique**(name, salary) overlap since the set of employee objects restricted by these constraints are overlapping. The same argument holds for the constraints **unique**(ssn, name) and **unique**(name, salary). □

Relationships between C4*-Constraints.* The relationship between C4-constraints is determined by the relationships between the antecedents and consequents of the implications. Let there be two C4-constraints IC_1 and IC_2 of the form $(\mathsf{X}_1 \Rightarrow \mathsf{Y}_1)$ and $(\mathsf{X}_2 \Rightarrow \mathsf{Y}_2)$, respectively, where $\mathsf{X}_1, \mathsf{X}_2, \mathsf{Y}_1$, and Y_2 are constraints of type C1 or C2. Then, we can derive rules of the following form to compute the relationship between C4-constraints:

$$(\mathsf{X}_1\, \Theta_1\, \mathsf{X}_2) \wedge (\mathsf{Y}_1\, \Theta_2\, \mathsf{Y}_2) \Rightarrow (IC_1\, \Theta_3\, IC_2)$$

The most combinations of Θ_1 and Θ_2 result in an overlap relationship between IC_1 and IC_2. The table beside summarizes the combinations which lead to another relationship than overlap. We omit the proofs of these rules. Instead, we present an example to demonstrate some of these rules.

Θ_1	Θ_2	Θ_3
$\equiv$	$\equiv$	$\equiv$
$\equiv$	$\supset$	$\supset$
$\subset$	$\equiv, \supset$	$\supset$
$\supset$	$\equiv, \subset$	$\subset$
$\equiv$	$\subset$	$\subset$

Example 4. Let there be the following integrity constraints sets:

$$\mathcal{IC}_1 = \{(\mathsf{salary} > 10000) \Rightarrow (\mathsf{age} > 35)\}$$
$$\mathcal{IC}_2 = \{(\mathsf{salary} < 5000) \Rightarrow (\mathsf{age} < 25)\}$$
$$\mathcal{IC}_3 = \{(\mathsf{salary} < 2000) \Rightarrow (\mathsf{age} < 40)\}$$

From the rules above follows that $\mathcal{IC}_1 \Cap \mathcal{IC}_2$ hold (because the antecedents as well as the consequents of the implications are "disjoint"). For example, a forty year old employee with a salary of 2000 is conform to $\mathcal{IC}_1$, but not to $\mathcal{IC}_2$. On the other hand, a twenty year old employee with a salary of 30000 is conform to $\mathcal{IC}_2$, but not to $\mathcal{IC}_1$. However, there are employees which are conform to both integrity constraints, e.g. a fifty year old employee with a salary of 15000.

Comparing $\mathcal{IC}_2$ with $\mathcal{IC}_3$ lead to a superset relationship $\mathcal{IC}_2 \supset \mathcal{IC}_3$ (see the third rule in table above where the antecedents are in a subset relationship and the consequents are in a subset relationship). That is, the integrity constraint $\mathcal{IC}_2$ is more restrictive than $\mathcal{IC}_3$. For instance, $\mathcal{IC}_3$ allows thirty year old employees with a salary of 4000, whereas $\mathcal{IC}_2$ does not. □

Relationships between Different Types of Constraints. Up to now, we have discussed relationships between constraints of the same type. We worked out a set of rules for computing the corresponding relationships. Investigating relationships between constraints of different types, we come to the conclusion that, as a rule, the only possible relationship is overlap. We will not formally prove this fact. Instead, we present some examples for illustration.

Example 5. Suppose, there are the following integrity constraint sets given:

$$\mathcal{IC}_1 = \{(\mathsf{salary} > 2000)\}$$
$$\mathcal{IC}_2 = \{(\mathsf{salary} > \mathsf{bonus})\}$$
$$\mathcal{IC}_3 = \{\textbf{unique}(\mathsf{salary})\}$$
$$\mathcal{IC}_4 = \{(\mathsf{salary} > 2000 \Rightarrow \mathsf{bonus} < 1000)\}$$

Comparing the sets of possible objects restricted by the constraints $\mathcal{IC}_1$ and $\mathcal{IC}_2$, we see that $\mathcal{IC}_1$ allows employees to have an arbitrary value for the attribute bonus — since there is no restriction on this attribute. Thus, an employee may have a "fixed" salary higher than 2000 and bonus "salary" higher than 1000. Such a value combination is forbidden by the constraint $\mathcal{IC}_2$. However, this constraint allows employees to have a salary less than 2000. Since both constraints also allow same value combinations for salary and bonus, e.g. (salary=3000, bonus=500), we can state that these constraints are overlapping.

When we compare the constraints $\mathcal{IC}_1$ and $\mathcal{IC}_3$, we obtain the result that $\mathcal{IC}_1$ allows two employees with the same salary (which must be higher than 2000), whereas this is forbidden by $\mathcal{IC}_3$. On the other hand, $\mathcal{IC}_3$ allows employees to have a salary less than 2000. Since both constraints also allow same values for the attribute salary, we derive that these constraints are overlapping.

Finally, let us have a closer look at the relationship between the constraints $\mathcal{IC}_2$ and $\mathcal{IC}_4$. The constraint $\mathcal{IC}_2$ states that the "fixed" salary of an employee have to be higher than his bonus "salary". Hence, an employee may have a "fixed" salary higher than 2000 and bonus "salary" higher than 1000, e.g. (salary=5000, bonus=2500). Such an attribute value combination is forbidden by the integrity constraint $\mathcal{IC}_4$. However, the constraint $\mathcal{IC}_4$ allows employees to have a bonus "salary" which higher than their "fixed" salary, if the "fixed" salary is less than 2000, e.g. (salary=1000, bonus=2500). Since both constraints also allow same attribute value combinations for salary and bonus, e.g. (salary=3000, bonus=500), we can state that these constraints are overlapping. □

In contrast, when we consider integrity constraint sets containing at least two constraints, further relationships are possible. Sometimes, it is possible to derive an implicit constraint of type C2 from two constraints of type C1. In the following, we investigate the relation between a set of two constraints of type C1 and a constraint of type C2. We start with an illustrating example.

Example 6. Let there be the following integrity constraint sets:

$$\mathcal{IC}_1 = \{(\mathsf{salary} > 2000), (\mathsf{bonus} < 1000)\}$$
$$\mathcal{IC}_2 = \{(\mathsf{salary} > \mathsf{bonus})\}$$

The first integrity constraint set states that all employees must have a "fixed" salary higher than 2000 and a bonus "salary" which does not exceed 1000. The second integrity constraint set says that the "fixed" salary of each employee must be higher than his bonus "salary". Obviously, the latter constraint is also implicitly expressed by the conjunction of the two constraints of IC_1. Thus, we can state that $\mathcal{IC}_1$ is stronger than $\mathcal{IC}_2$, i.e. that $\mathcal{IC}_1 \supset \mathcal{IC}_2$ holds. □

Motivated by the example above, we have analyzed all combinations of comparison operators and constant relationships (of two $\mathsf{C1}$-constraints). As a result, we found a set rules of the form

$$(\mathsf{x}\,\theta_1\,\mathsf{c}_1) \wedge (\mathsf{y}\,\theta_2\,\mathsf{c}_2) \wedge (\mathsf{c}_1\,\theta_3\,\mathsf{c}_2) \Rightarrow (\mathsf{x}\,\theta_4\,\mathsf{y})$$

to derive an implicit $\mathsf{C2}$-constraint from two $\mathsf{C1}$-constraints. We have to point out that not all combinations of the comparison operators θ_1, θ_2, and θ_3 lead to an implicit $\mathsf{C2}$-constraint. Fig. 3 depicts the combinations which lead to an implicit constraint. Furthermore, there are some cases where a relationship between an

$(\mathsf{x}\,\theta_1\,\mathsf{c}_1) \wedge (\mathsf{y}\,\theta_2\,\mathsf{c}_2) \wedge (\mathsf{c}_1\,\theta_3\,\mathsf{c}_2) \Rightarrow (\mathsf{x}\,\theta_4\,\mathsf{y})$			
θ_1	θ_2	θ_3	θ_4
$<$	$=, \geq, >$	$<, =$	$<$
$\leq, =$	$=, \geq, >$	$<$	$<$
$\leq, =$	$>$	$=$	$<$
$\leq$	$=, \geq$	$=$	$\leq$
$=$	$\geq$	$=$	$\leq$
$=$	$=$	$=$	$=$
$=$	$\neq$	$=$	$\neq$
$=$	$\leq$	$=$	$\geq$
$=$	$<, \leq, =$	$>$	$>$
$=$	$<$	$=$	$>$

Fig. 3. Rules for Deriving Implicit $\mathsf{C2}$-Constraints

integrity constraint set with (at least) two constraints of type $\mathsf{C1}$ (or $\mathsf{C2}$) and a $\mathsf{C4}$-constraint exists. An $\mathsf{C4}$-constraint $(\mathsf{x} \Rightarrow \mathsf{y})$ can be expressed as $(\neg\mathsf{x} \vee \mathsf{y})$. Since we know that generally

$$(IC_1 \wedge IC_2) \Rightarrow (IC_1 \vee IC_2)$$

holds, we can derive the following rules for computing relationships between an integrity constraint set $\mathcal{IC}_1$ with two constraints x_1 and y_1 and an integrity constant set $\mathcal{IC}_2$ with an $\mathsf{C4}$-constraint of the form $(\mathsf{x}_2 \Rightarrow \mathsf{y}_2)$:

$$(\mathcal{IC}_1 \supset \mathcal{IC}_2) \Leftrightarrow (\mathsf{x}_1\,\Theta_1\,\mathsf{x}_2) \wedge (\mathsf{y}_1\,\Theta_2\,\mathsf{y}_2),\ \Theta_1, \Theta_2 \in \{\equiv, \subset\}$$

All other combinations of Θ_1 and Θ_2 lead to an overlap relationship between $\mathcal{IC}_1$ and $\mathcal{IC}_2$.

Example 7. Suppose the following two integrity constraint sets are given:

$$\mathcal{IC}_1 = \{(\mathsf{salary} > 2000), (\mathsf{bonus} < 1000)\}$$
$$\mathcal{IC}_2 = \{(\mathsf{salary} \leq 2000) \Rightarrow (\mathsf{bonus} < 1000)\}$$

Then, we can state that $\mathcal{IC}_1 \supset \mathcal{IC}_2$ holds, since $\mathcal{IC}_1$ implies $\mathcal{IC}_2$. □

Deriving Relationships between Integrity Constraint Sets. Up to now, our discussion was restricted to single integrity constraints or integrity constraint sets with specific constraint types. In the following, we extend our considerations to general integrity constraint sets which may contain constraints of different types. The following theorem provides the rules for computing the relationship between two general integrity constraint sets.

Theorem 2 (Relationship between Integrity Constraint Sets). For two extended integrity constraint sets $\mathcal{IC}_1$ and $\mathcal{IC}_2$ the following rules hold:

$$
\begin{aligned}
(\mathcal{IC}_1 \Cup \mathcal{IC}_2) &:\Leftrightarrow (\forall IC_{1i} \in \mathcal{IC}_1(\nexists IC_{2j} \in \mathcal{IC}_2(IC_{1i}\ \vartheta\ IC_{2j}),\ \vartheta \in \{\varnothing, \equiv, \subset, \supset, \Cap\}) \\
(\mathcal{IC}_1 \varnothing \mathcal{IC}_2) &:\Leftrightarrow (\exists IC_{1i} \in \mathcal{IC}_1(\exists IC_{2j} \in \mathcal{IC}_2((IC_{1i} \varnothing IC_{2j})))) \\
(\mathcal{IC}_1 \equiv \mathcal{IC}_2) &:\Leftrightarrow ((\forall IC_{1i} \in \mathcal{IC}_1(\exists IC_{2j} \in \mathcal{IC}_2((IC_{1i} \equiv IC_{2j}))) \wedge \\
&\qquad (\forall IC_{2j} \in \mathcal{IC}_2(\exists IC_{1i} \in \mathcal{IC}_1((IC_{1i} \equiv IC_{2j})))) \\
(\mathcal{IC}_1 \supset \mathcal{IC}_2) &:\Leftrightarrow (\forall IC_{2j} \in \mathcal{IC}_2(\exists IC_{1i} \in \mathcal{IC}_1((IC_{1i} \equiv IC_{2j}) \vee (IC_{1i} \supset IC_{2j}))) \wedge \\
&\qquad (\exists IC_{2j} \in \mathcal{IC}_2(\exists IC_{1i} \in \mathcal{IC}_1((IC_{1i} \supset IC_{2j})))) \\
(\mathcal{IC}_1 \Cap \mathcal{IC}_2) &:\Leftrightarrow \neg((\mathcal{IC}_1 \Cup \mathcal{IC}_2) \vee (\mathcal{IC}_1 \varnothing \mathcal{IC}_2) \vee (\mathcal{IC}_1 \equiv \mathcal{IC}_2) \vee (\mathcal{IC}_1 \subset \mathcal{IC}_2) \vee \\
&\qquad\quad (\mathcal{IC}_1 \supset \mathcal{IC}_2))
\end{aligned}
$$

Please note that $IC \supset \textsf{\textbf{true}}$ always holds, if a corresponding related constraint does not exist in the other integrity constraint set. □

Proof. In the following, we exemplary prove the "disjoint" rule. Since the proofs of the other rules can be shown analogously, we omit the proof of these rules in order to keep the paper concise.

The integrity constraint sets $\mathcal{IC}_1$ and $\mathcal{IC}_2$ can be represented as a conjunction of logical formulas where each term corresponds to an integrity constraint:

$$
\begin{aligned}
\mathcal{IC}_1 &\rightarrow IC_{11} \wedge \ldots \wedge IC_{1n} \\
\mathcal{IC}_2 &\rightarrow IC_{21} \wedge \ldots \wedge IC_{2m}
\end{aligned}
$$

Since we assume that each integrity constraint set is consistent, the conjunction of the logical representation of the integrity constraint sets implies false if and only if there exists at least one pair of terms whose conjunction implies false:

$$
\begin{aligned}
&((IC_{11} \wedge \ldots \wedge IC_{1n}) \wedge (IC_{21} \wedge \ldots \wedge IC_{2m}) \Rightarrow \textsf{\textbf{false}}) \\
\Leftrightarrow\ &(\exists IC_{1i}(\exists IC_{2j}(IC_{1i} \wedge IC_{2j} \Rightarrow \textsf{\textbf{false}}))) \\
\leftrightarrow\ &(\exists IC_{1i} \in \mathcal{IC}_1(\exists IC_{2j} \in \mathcal{IC}_2((IC_{1i} \varnothing IC_{2j})))) \\
\Leftrightarrow\ &(\mathcal{IC}_1 \varnothing \mathcal{IC}_2)
\end{aligned}
$$

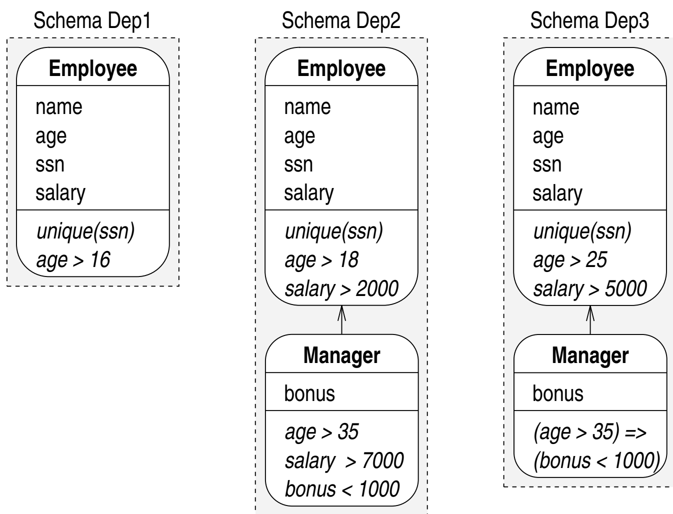
Thus, we can conclude that the disjointness of two integrity constraints implies the disjointness of the corresponding integrity constraint sets. □

Theorem 3. The relationship between integrity constraint sets containing constraints of type $\mathsf{C1}$, $\mathsf{C2}$, $\mathsf{C3}$, and/or $\mathsf{C4}$ is computable in polynomial time. □

Proof. The rules can be realized by an algorithm consisting of two loops where each pair of constraints of the different sets are compared. Let the cardinality of the integrity constraints sets be n and m, respectively. Then, the cardinality of the extended integrity constraint sets is n^2 and m^2, respectively, in the worst case (for our constraint types). Hence, at most $n^2 \times m^2$ comparison operations are needed to determine the relationship between integrity constraint sets. □

3 Comparing Classes of Different Schemata

Suppose, the following three example schemata are given:



In the sequel, we relate the classes of these schemata using the rules previously introduced. First, we derive the integrity constraint sets of all classes existing in the three example schemata:

$$\begin{aligned}
\mathcal{IC}_{\mathsf{Dep1.Emp}} &= \{\textbf{unique}(\mathsf{ssn}), (\mathsf{age} > 16)\} \\
\mathcal{IC}_{\mathsf{Dep2.Emp}} &= \{\textbf{unique}(\mathsf{ssn}), (\mathsf{age} > 18), (\mathsf{salary} > 2000)\} \\
\mathcal{IC}_{\mathsf{Dep2.Mng}} &= \{\textbf{unique}(\mathsf{ssn}), (\mathsf{age} > 35), (\mathsf{salary} > 7000), (\mathsf{bonus} < 1000)\} \\
\mathcal{IC}_{\mathsf{Dep3.Emp}} &= \{\textbf{unique}(\mathsf{ssn}), (\mathsf{age} > 25), (\mathsf{salary} > 5000)\} \\
\mathcal{IC}_{\mathsf{Dep3.Mng}} &= \{\textbf{unique}(\mathsf{ssn}), (\mathsf{age} > 25), (\mathsf{salary} > 5000), \\
&\qquad (\mathsf{age} > 35) \Rightarrow (\mathsf{bonus} < 1000)\}
\end{aligned}$$

By definition of the class specialization concept the following relationships holds:

$$\begin{aligned}
\mathcal{IC}_{\mathsf{Dep2.Emp}} &\subset \mathcal{IC}_{\mathsf{Dep2.Mng}} \\
\mathcal{IC}_{\mathsf{Dep3.Emp}} &\subset \mathcal{IC}_{\mathsf{Dep3.Mng}}
\end{aligned}$$

Comparing the integrity constraint sets of the employee classes leads to:

$$\mathcal{IC}_{\mathsf{Dep1.Emp}} \subset \mathcal{IC}_{\mathsf{Dep2.Emp}}$$
$$\mathcal{IC}_{\mathsf{Dep1.Emp}} \subset \mathcal{IC}_{\mathsf{Dep3.Emp}}$$
$$\mathcal{IC}_{\mathsf{Dep2.Emp}} \subset \mathcal{IC}_{\mathsf{Dep3.Emp}}$$

Below we exemplary sketch the derivation of the first relationship above:

1. Take the first integrity constraint in $\mathcal{IC}_{\mathsf{Dep1.Emp}}$ and search for a related constraint in $\mathcal{IC}_{\mathsf{Dep2.Emp}}$. Since the constraint **unique**(ssn) is in both integrity constraints sets, notice that there is an equivalent constraint:

$$\textbf{unique}(\mathsf{ssn}) \equiv \textbf{unique}(\mathsf{ssn})$$

2. When we apply the same procedure on the second constraint of $\mathcal{IC}_{\mathsf{Dep1.Emp}}$, we obtain the following relationship:

$$(\mathsf{age} > 16) \subset (\mathsf{age} > 18)$$

3. Since for all constraints in $\mathcal{IC}_{\mathsf{Dep1.Emp}}$ there exists a related constraint in $\mathcal{IC}_{\mathsf{Dep2.Emp}}$ which is equivalent or stronger, we can derive according to Theorem 2 the following relationship for the given integrity constraint sets:

$$\mathcal{IC}_{\mathsf{Dep1.Emp}} \subset \mathcal{IC}_{\mathsf{Dep2.Emp}}$$

In summary, we imply that class Employee of schema Dep1 is a superclass of class Employee of schema Dep2. The class Employee of schema Dep2, on the other hand, is a superclass of class Employee of schema Dep3. Thus, class Employee of schema Dep1 is transitively a superclass of class Employee of schema Dep3.

Comparing the integrity constraint sets of the manager classes, we obtain the relationship $\mathcal{IC}_{\mathsf{Dep2.Mng}} \supset \mathcal{IC}_{\mathsf{Dep3.Mng}}$:

$$\textbf{unique}(\mathsf{ssn}) \equiv \textbf{unique}(\mathsf{ssn})$$
$$(\mathsf{age} > 35) \supset (\mathsf{age} > 25)$$
$$(\mathsf{salary} > 7000) \supset (\mathsf{salary} > 5000)$$
$$(\mathsf{bonus} < 1000) \supset \textbf{true}$$
$$\{(\mathsf{age} > 35), (\mathsf{bonus} < 1000)\} \supset \{(\mathsf{age} > 35) \Rightarrow (\mathsf{bonus} < 1000)\}$$

For the first relationship see Theorem 1. The second and third relationships are derived using the negated result of rule 11 in Fig. 1. The fourth relationship is given by definition. The fifth relationship is computed by the rule given at Page 8 (see also Example 7). From the relationship $\mathcal{IC}_{\mathsf{Dep2.Mng}} \supset \mathcal{IC}_{\mathsf{Dep3.Mng}}$ immediately follows that the class Manager of schema Dep2 is a subclass of the class Manager of schema Dep3. In consequence, the class Manager of schema

Dep2 is transitively a subclass of the other classes. In conclusion, the extensional relationships among the classes of the example schemata are as follows:

$$\mathsf{Dep1.Emp} \supset \mathsf{Dep2.Emp} \supset \mathsf{Dep3.Emp} \supset \mathsf{Dep3.Mng} \supset \mathsf{Dep2.Mng}$$

These extensional relationships hold if all extensional restrictions are expressed by integrity constraints. Since extensional relationships are used as basic input information for a schema integration algorithm, we can state that the quality of the integrated schema (which should be conform to the modeled real world) also depends on the quality of the result of the schema comparison.

4 Conclusions

In this paper, we worked out the correspondence between integrity constraints as foundation for semantic schema comparison. We pointed out the relevance of a correct schema comparison as basis for a semantically correct schema integration. In particular, we have presented rules for computing the relationship between integrity constraint sets consisting certain types of constraints. These rules can be used to derive extensional relationships between classes of different schemata.

Currently, we are investigating how far rules for relating complex integrity constraints can be derived. There are also relationships between aggregation constraints and other types of integrity constraints. For instance, from the constraint ($\mathsf{salary} > 2000$) we can derive the constraint ($\mathbf{avg}(\mathsf{salary}) > 2000$). In this case, the first constraint implies the second one.

Acknowledgments: We thank Kerstin Schwarz for useful hints. This work was partly supported by the German Federal State Sachsen-Anhalt under FKZ 1987/2527R.

References

1. C. Batini, M. Lenzerini, S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, 1986.
2. Y. Dupont, S. Spaccapietra. Schema Integration Engineering in Cooperative Databases Systems. In [8], pp. 759–765.
3. M. Garcia-Solaco, M. Castellanos, F. Saltor. A Semantic-Discriminated Approach to Integration in Federated Databases. In S. Laufmann, S. Spaccapietra, T. Yokoi (eds.), *Proc. CoopIS'95*, pp. 19–31, 1995.
4. S. B. Navathe, R. Elmasri, J. A. Larson. Integrating User Views in Database Design. *IEEE Computer*, 19(1):50–62, 1986.
5. I. Schmitt, G. Saake. Schema Integration and View Generation by Resolving Intensional and Extensional Overlappings. In [8], pp. 751–758.
6. A. P. Sheth, J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, 1990.
7. S. Spaccapietra, C. Parent, Y. Dupont. Model Independent Assertions for Integration of Heterogeneous Schemas. *The VLDB Journal*, 1(1):81–126, 1992.
8. K. Yetongnon, S. Hariri (eds.). *Proc. 9th ISCA Int. Conf. on Parallel and Distributed Computing Systems, PDCS'96*, International Society for Computers and Their Application, Six Forks Road, Releigh, NC, 1996.