Acta Informatica manuscript No.
(will be inserted by the editor)

On the complexity of deciding typability in the relational algebra

Stijn Vansummeren[⋆]

Limburgs Universitair Centrum, Diepenbeek, Belgium
e-mail: stijn.vansummeren@luc.ac.be

The date of receipt and acceptance will be inserted by the editor

Abstract. We investigate the complexity of the typability problem for the relational algebra. This problem consists of deciding, for a given relational algebra expression, whether there exists an assignment of types to variables occurring in the expression such that the expression is well-typed under the assignment. We obtain that the problem is NP-complete in general. In particular, we show that the problem becomes NP-hard due to (1) the cartesian product operator; (2) the selection operator on arbitrary sets of typed predicates; and (3) the selection operator on "well-behaved" sets of typed predicates together with join and projection or renaming. However, the problem is in P when (1) we only allow union, difference, join and selection on "well-behaved" sets of typed predicates; or (2) we allow all operators except cartesian product, where the set of selection predicates can mention at most one base type. Most of these results follow from a close connection of the typability problem to non-uniform constraint satisfaction.

1 Introduction

The relational algebra is the basis of all relational query languages [1,2]. For a relational algebra expression to be well-defined, its relation variables must satisfy certain requirements. For example, in order for the expression $\sigma_{A=5}(r)$ to have any meaning, relation r needs to have an integer-valued A attribute. In the theory of programming languages [5], such requirements are recorded in *types* and *type systems* are used to discern ill-defined expressions from well-defined ones. In particular, well-defined expressions

⋆ Research Assistant of the Fund for Scientific Research - Flanders (Belgium)

are those expressions that can be assigned a type in the system. Recently, Van den Bussche and Waller [7] considered a type system for the relational algebra. This system consists of a set of *type rules* allowing to derive the output type of a program given types for its relation variables. If such an output type can be derived for a given type assignment, the expression is called *well-typed* under the assignment. Van den Bussche and Waller developed a *type inference* algorithm capable of determining the *principal type formula* of a given expression. This formula describes all assignments of types to relation names, under which a given relational algebra expression is well-typed, as well as the output type that expression will have under each of these assignments.

For general motivations for studying type inference for the relational algebra, we refer to Van den Bussche and Waller [7]. In this paper, we will focus on *typability*: deciding whether a given expression can ever be well-defined; that is, does there exist a type assignment for which the expression is well-typed? Traditional type inference algorithms also check typability (i.e. they return *false* instead of a principal type formula when an expression is not typable). By studying the complexity of the typability problem, we hence get an insight into the complexity of the corresponding type inference problem.

Van den Bussche and Waller showed that typability for the relational algebra is decidable in non-deterministic polynomial time. The precise complexity remained open. In this paper we obtain that the problem is NP-complete in general. In particular, we show that the problem becomes NP-hard due to (1) the cartesian product operator; (2) the selection operator on arbitrary sets of typed predicates; and (3) the selection operator on "well-behaved" sets of typed predicates together with join and projection or renaming. However, the problem is in P when (1) we only allow union, difference, join and selection on "well-behaved" sets of typed predicates; or (2) we allow all operators except cartesian product, where the set of selection predicates can mention at most one base type. Most of these results follow from a close connection of the typability problem to non-uniform constraint satisfaction.

This paper is further organized as follows. We introduce the type system of Van den Bussche and Waller in Section 2, including the notions of well-typedness and typability. We then show that the typability problem is NP-complete in its most general setting in Section 3. In Section 4 we illustrate the close connection between the typability problem and constraint satisfaction problems, which allows us to obtain most of the results mentioned above. We summarize our results in Section 5.

2 Preliminaries

The type system of Van den Bussche and Waller [7] considers an expression ill-defined if the schema of some subexpression is required to have an attribute present and absent at the same time. Practical query languages also consider an expression ill-defined if attributes are not used consistently with regard to their base types. For instance, the relational algebra expression $\sigma_{A=5}(\sigma_{A=true}(R))$ would be considered correct in the framework of Van den Bussche and Waller. The corresponding SQL expression would be considered incorrect however, since an attribute cannot have base type `bool` and `int` at the same time. In this section, we therefore augment the type system of Van den Bussche and Waller [7] with the ability to deal with types on the attribute value level.

We assume given sufficiently large sets of *relation variables* and *attribute names*. Relation variables will be denoted by lowercase letters from the end of the alphabet, whereas attribute names will be denoted by uppercase letters from the beginning of the alphabet.

We also assume to be given a finite, non-empty set of *base types* (such as $\texttt{int}, \texttt{string}, \texttt{bool}, \ldots$) and a sufficiently large set of *predicates* (such as $=, \leq, \ldots$). Every predicate θ has an *arity* $|\theta|$, which is a natural number, and a *signature* $\varsigma(\theta)$, which is a non-empty, $|\theta|$-ary relation over the set of base types. An example of a signature for the binary predicate has_length is $\{(\texttt{string}, \texttt{int})\}$, while an example of a signature for the binary predicate $=$ is

$$\{(\texttt{int}, \texttt{int}), (\texttt{bool}, \texttt{bool}), (\texttt{string}, \texttt{string})\}.$$

Base types will be denoted by β, possibly subscripted. Predicates will be denoted by θ, possibly subscripted. Finite sets of predicates will be denoted by Θ. The set of all base types mentioned in signatures of predicates in Θ will be denoted by $\beta(\Theta)$.

The relational algebra with selection predicates in Θ, denoted by $\mathcal{R}^{\Theta}$, is the set of all expressions generated by the following grammar:

$$\begin{aligned} e \rightarrow\ & r \\ \mid\ & (e \cup e) \mid (e - e) \mid (e \bowtie e) \mid (e \times e) \\ \mid\ & \sigma_{\theta(A_1,\ldots,A_n)}(e) \mid \pi_{A_1,\ldots,A_n}(e) \mid \rho_{A/B}(e) \end{aligned}$$

Here e denotes an expression, r denotes a relation variable, θ denotes a selection predicate in Θ of arity n, and A, B, and A_i denote attribute names. If V is a set of operators, then $\mathcal{R}^{\Theta}_{V}$ denotes the subset of expressions in $\mathcal{R}^{\Theta}$ using only operators in V. The semantics of relational algebra is the well-known one [1,2] and will actually not concern us in the present paper. The set of all relation variables occurring in expression e is denoted

$$\frac{\Gamma(r) = \tau}{\Gamma \vdash r : \tau} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \cup e_2 : \tau} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 - e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \sim \tau_2}{\Gamma \vdash e_1 \bowtie e_2 : \tau_1 \cup \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \mathit{dom}(\tau_1) \cap \mathit{dom}(\tau_2) = \emptyset}{\Gamma \vdash e_1 \times e_2 : \tau_1 \cup \tau_2}$$

$$\frac{\Gamma \vdash e : \tau \quad A_1, \ldots, A_n \in \mathit{dom}(\tau) \quad (\tau(A_1), \ldots, \tau(A_n)) \in \varsigma(\theta)}{\Gamma \vdash \sigma_{\theta(A_1,\ldots,A_n)}(e) : \tau} \qquad \frac{\Gamma \vdash e : \tau \quad A_1, \ldots, A_n \in \mathit{dom}(\tau)}{\Gamma \vdash \pi_{A_1,\ldots,A_n}(e) : \{A_1, \ldots, A_n\}}$$

$$\frac{\Gamma \vdash e : \tau \quad A \in \mathit{dom}(\tau) \quad B \notin \mathit{dom}(\tau)}{\Gamma \vdash \rho_{A/B}(e) : \rho_{A/B}(\tau)}$$

Fig. 1 The typing relation

by *Relvars*(e) and the set of all attributes occurring in e is denoted by *Specattrs*(e).

A *type* τ is a function from a finite set of attribute names *dom*(τ) to the set of base types. Two types are *compatible*, denoted by $\tau_1 \sim \tau_2$, if $\tau_1(A) = \tau_2(A)$ for each A in *dom*$(\tau_1) \cap$ *dom*(τ_2). Clearly, the union of two compatible types is defined, and is again a type. If τ is a type, then $\rho_{A/B}(\tau)$ is the type with domain *dom*$(\tau) - \{A\} \cup \{B\}$ such that $\rho_{A/B}(\tau)(B) = \tau(A)$ and $\rho_{A/B}(\tau)(C) = \tau(C)$ for every $C \in$ *dom*$(\tau) - \{A, B\}$. A *schema* is a finite set $\mathcal{S}$ of relation variables. A *type assignment* on $\mathcal{S}$ is a mapping Γ on $\mathcal{S}$, assigning to each $r \in \mathcal{S}$ a type $\Gamma(r)$.

Let e be an expression in $\mathcal{R}^{\Theta}$, $\mathcal{S}$ a schema with *Relvars*$(e) \subseteq \mathcal{S}$, and Γ a type assignment on $\mathcal{S}$. Then the relation $\Gamma \vdash e : \tau$, indicating that e *has type* τ *under* Γ, is defined by the rules in Figure 1. If $A \in$ *dom*(τ), then we say that A is *present* in e under Γ. We say that A is *absent* in e under Γ otherwise. Note that e has at most one type under Γ, which can easily be derived from Γ by applying the rules in an order determined by the syntax of expression e.

The central notion of this paper is given by the following definition:

Definition 1. *Let e be an expression in $\mathcal{R}^{\Theta}$ and let Γ be a type assignment on Relvars(e). If there exists a type τ such that $\Gamma \vdash e : \tau$, we say that e is* well-typed *under Γ. Expression e is called* typable *if there exists a type assignment Γ on Relvars(e) such that e is well-typed under Γ.*

Let V be a subset of the relational algebra operators. We denote the set of all typable expressions in $\mathcal{R}^{\Theta}_V$ by $\mathcal{T}(\mathcal{R}^{\Theta}_V)$. Deciding membership of $\mathcal{T}(\mathcal{R}^{\Theta}_V)$ is called the *typability problem*.

3 Deciding typability

Van den Bussche and Waller noted that the typability problem can be solved in non-deterministic polynomial time [7]. We reiterate their result here for completeness' sake. If τ is a type and $\mathcal{A}$ is a set of attribute names, then we write $\tau|_{\mathcal{A}}$ for the type defined by $\tau|_{\mathcal{A}}(A) := \tau(A)$ for every A in $\mathit{dom}(\tau) \cap \mathcal{A}$. If Γ is a type assignment, then we write $\Gamma|_{\mathcal{A}}$ for the type assignment defined by $\Gamma|_{\mathcal{A}}(r) := \Gamma(r)|_{\mathcal{A}}$.

Lemma 2 (Van den Bussche and Waller). *If $\Gamma \vdash e : \tau$ and $\mathit{Specattrs}(e) \subseteq \mathcal{A}$, then $\Gamma|_{\mathcal{A}} \vdash e : \tau|_{\mathcal{A}}$.*

The proof is straightforward. As a consequence, in order to decide whether there exists a type assignment under which e is well-typed, it suffices to consider type assignments Γ with the property that

$$\mathit{dom}(\Gamma(r)) \subseteq \mathit{Specattrs}(e),$$

for every $r \in \mathit{Relvars}(e)$. It follows immediately that typability is in NP. This upper bound is tight, as the following theorem shows.

Theorem 3. *$\mathcal{T}(\mathcal{R}^{\Theta})$ is* NP*-complete for any predicate-set Θ.*

Proof. We give a LOGSPACE reduction from POSITIVE ONE-IN-THREE 3SAT, which is known to be NP-hard [3]. The POSITIVE ONE-IN-THREE 3SAT problem consists of deciding for a given 3CNF formula with only positive clauses of the form $(x \vee y \vee z)$, whether there exists a truth assignment that makes exactly one literal per clause true.

Let $\phi = (x_1 \vee y_1 \vee z_1) \wedge \cdots \wedge (x_n \vee y_n \vee z_n)$ be a 3CNF formula where every clause is positive. Let X be the set of all variables occurring in ϕ. We construct the expression e_ϕ using X as relation variables, such that ϕ is one-in-three satisfiable if, and only if, e_ϕ is typable:

$$e_\phi := \bigcup_{i=1}^{n} \pi_A(x_i \times y_i \times z_i).$$

It is clear that e_ϕ can be constructed from ϕ in logarithmic space.

Suppose ϕ is one-in-three satisfiable. Then there exists a truth assignment w on X such that for every i exactly one of $w(x_i)$, $w(y_i)$, and $w(z_i)$ is true. To show that e_ϕ is typable, we construct the type assignment Γ on X as follows. Let τ be a type with domain $\{A\}$. We define $\Gamma(x) := \tau$ if $w(x)$ is true, and $\Gamma(x) := \emptyset$ otherwise. Since exactly one of $w(x_i)$, $w(y_i)$ and $w(z_i)$ is true for every i, we have by construction that exactly one of $\Gamma(x_i)$, $\Gamma(y_i)$, and $\Gamma(z_i)$ is τ, the others being $\emptyset$. Since the domains of $\Gamma(x_i)$, $\Gamma(y_i)$

and $\Gamma(z_i)$ are then disjoint and since $\Gamma(x_i) \cup \Gamma(y_i) \cup \Gamma(z_i) = \tau$, the expressions $x_i \times y_i \times z_i$ have type τ under Γ. Then every $\pi_A(x_i \times y_i \times z_i)$ also has type τ under Γ. Therefore every operand of the union operator has type τ, and thus e_ϕ is well-typed under Γ.

Conversely, suppose e_ϕ is typable. Then there exists a type assignment Γ on X such that every subexpression of e (including e) is well-typed under Γ. To show that ϕ is satisfiable, we construct the truth assignment w on X such that $w(u)$ is true if, and only if, $A \in \mathit{dom}(\Gamma(u))$. Since every $\pi_A(x_i \times y_i \times z_i)$ is well-typed under Γ, the type of $x_i \times y_i \times z_i$ must be defined on A. Because of the typing rule for $\times$, this means that at exactly one of $\Gamma(x_i), \Gamma(y_i)$, and $\Gamma(z_i)$ is defined on A. Hence, exactly one of $w(x_i), w(y_i)$, or $w(z_i)$ is true for every i, and ϕ is one-in-three satisfiable. □

The following natural question now arises: for which operators of the relational algebra can typability be decided in polynomial time? We note that expressions in $\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\times}$ are always typable, hence $\mathcal{T}(\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\times})$ is trivially in P. However, adding π, ρ, or σ to the set of operators immediately makes the problem NP-complete. This is clear for π from the reduction above. Also, the reduction still works if we define e_ϕ as

$$e_\phi := \bigtimes_{i=1}^{n} \rho_{A/B_i}(x_i \times y_i \times z_i).$$

Here the B_i are auxiliary attribute names used to make sure that the various operands of $\times$ have a disjoint domain. Finally, if $\theta \in \Theta$, then we can define e_ϕ as:

$$e_\phi := \bigcup_{i=1}^{n} \sigma_{\theta(A_1,\ldots,A_k)}(x_i \times y_i \times z_i).$$

Indeed, if ϕ is one-in-three satisfiable, then we can show typability of e_ϕ simply by taking τ in the reasoning above to be a type for which

$$(\tau(A_1), \ldots, \tau(A_k)) \in \varsigma(\theta).$$

Conversely, if e_ϕ is typable, we can show one-in-three satisfiability of ϕ by taking w in the reasoning above such that $w(u)$ is true if, and only if, $A_1 \in \mathit{dom}(\Gamma(u))$.

Hence, deciding typability for restrictions of the relational algebra containing $\{-, \times, \pi\}$, $\{-, \times, \sigma\}$, $\{\cup, \times, \sigma\}$, or $\{\times, \rho\}$ as a subset of operators remains NP-complete for any (non-empty) predicate-set Θ.

4 Typability and constraint satisfaction

The results of Section 3 seem to imply that the cartesian product operator is the main reason why the typability problem is NP-hard. Consider expressions of the following form however:

$$\sigma_{\theta_1(A_1,\ldots,A_k)} \cdots \sigma_{\theta_n(B_1,\ldots,B_l)}(R).$$

In order to decide typability of such expressions, we need to make sure that there are no base-type clashes between the various uses of an attribute. It is not hard to see that this is another potential source of intractability.

We will formalize this intuition by showing that typability in $\mathcal{R}_\sigma^\Theta$ is a disguised form of the *non-uniform constraint satisfaction problem*, which is known to be NP-complete in general.

A *relational structure* is a tuple $(C, R_1, \ldots, R_n)$ where C is a finite set and $R_1, \ldots, R_n$ are relations over C. Let $\mathbf{A} = (C, R_1, \ldots, R_n)$ and $\mathbf{B} = (D, S_1, \ldots, S_n)$ be two relational structures where the arity of R_i equals the arity of S_i for every $1 \leq i \leq n$. A *homomorphism* from $\mathbf{A}$ to $\mathbf{B}$ is a function h from C to D such that for $1 \leq i \leq n$:

$$(c_1, \ldots, c_{k_i}) \in R_i \Rightarrow (h(c_1), \ldots, h(c_{k_i})) \in S_i.$$

Here, k_i denotes the arity of R_i and S_i.

The *constraint satisfaction problem* consists of deciding, given relational structures $\mathbf{A}$ and $\mathbf{B}$ whether there is a homomorphism from $\mathbf{A}$ to $\mathbf{B}$. This problem is NP-complete in general, since it is clearly in NP and it contains NP-hard problems as special cases. For example, 3-COLORABILITY is equivalent to the problem of deciding whether there is a homomorphism from a given graph $\mathbf{H}$ to the complete graph with 3 nodes

$$\mathbf{K_3} = (\{r, g, b\}, \{(r, b), (b, r), (r, g), (g, r), (b, g), (g, b)\}).$$

The constraint satisfaction problem for which $\mathbf{B}$ is fixed is called *the non-uniform constraint satisfaction problem* (non-uniform CSP). Let us define $\mathcal{H}(\mathbf{B})$ as the set of all structures for which there exists a homomorphism to $\mathbf{B}$. It is well-known that there exist structures $\mathbf{B}$ such that $\mathcal{H}(\mathbf{B})$ is NP-complete ($\mathbf{K_3}$ being an example).

We will now relate the typability problem to non-uniform CSP. Let $\Theta = \{\theta_1, \ldots, \theta_n\}$ be a set of predicates. We define the *structure* of an expression $e \in \mathcal{R}^\Theta$, denoted by *Struc*(e), as the relational structure

$$(\textit{Specattrs}(e), \theta_1(e), \ldots, \theta_n(e)),$$

where

$$\theta_i(e) = \{(A_1, \ldots, A_{|\theta_i|}) \mid \sigma_{\theta_i(A_1,\ldots,A_{|\theta_i|})}(e') \text{ is a subexpression of } e\}.$$

Likewise, we define the *structure of* Θ, denoted by *Struc*(Θ), as the relational structure $(\beta(\Theta), \varsigma(\theta_1), \ldots, \varsigma(\theta_n))$. Here, $\beta(\Theta)$ denotes the set of all base types mentioned in the signatures of predicates in Θ.

Lemma 4. *If Θ is a finite set of predicates and $e \in \mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma}$, then e is typable if, and only if, there is a homomorphism from Struc(e) to Struc(Θ).*

Proof. Intuitively, we need to make sure that there are no base-type clashes between the various uses of an attribute in e in order to decide typability of e. This is exactly what the existence of a homomorphism from *Struc*(e) to *Struc*(Θ) indicates.

Formally, suppose that there is a homomorphism h from *Struc*(e) to *Struc*(Θ). Let Γ be the type assignment defined by $\Gamma(r) = h$ for every $r \in$ *Relvars*(e). In order to show that e is typable, it suffices that to show that every subexpression e' of e has type h under Γ. We do so by induction on e'.

- Clearly, $\Gamma \vdash r : h$.
- If $e' = e_1 \cup e_2$ or $e' = e_1 - e_2$, then $\Gamma \vdash e_1 : h$ and $\Gamma \vdash e_2 : h$ by the induction hypothesis. Hence, all the premises of the type rule for union, respectively difference, are met and $\Gamma \vdash e' : h$ holds.
- If $e' = e_1 \bowtie e_2$, then then $\Gamma \vdash e_1 : h$ and $\Gamma \vdash e_2 : h$ by the induction hypothesis. Clearly, $h \sim h$ and $h = h \cup h$. Hence, all the premises of the type rule for join are met and $\Gamma \vdash e' : h$ holds.
- If $e' = \sigma_{\theta_i(A_1,\ldots,A_k)}(e'')$, then we have by the induction hypothesis that $\Gamma \vdash e'' : h$. By definition, $(A_1, \ldots, A_k) \in \theta_i(e)$. Since $\{A_1, \ldots, A_k\} \subseteq$ *Specattrs*(e) and since h is a homomorphism from *Struc*(e) to *Struc*(Θ) we have $(h(A_1), \ldots, h(A_k)) \in \varsigma(\theta_i)$. Hence, all the premises of the type rule for selection are met and $\Gamma \vdash e' : h$ holds.

Conversely, suppose that there exists a type assignment Γ under which e is well-typed. Let us write τ_e for the type of e under Γ and let us write $H(e)$ for the set of homomorphisms from *Struc*(e) to *Struc*(Θ). We prove by induction on e that $\tau_e|_{\textit{Specattrs}(e)} \in H(e)$.

- This is clear if $e = r$.
- If $e = e_1 \cup e_2$, then $\tau_e = \tau_{e_1} = \tau_{e_2}$ by the type rule for union. Then $\tau_e|_{\textit{Specattrs}(e_1)} \in H(e_1)$ and $\tau_e|_{\textit{Specattrs}(e_2)} \in H(e_2)$ by the induction hypothesis. Since *Struc*$(e) =$ *Struc*$(e_1) \cup$ *Struc*(e_2), it follows that $\tau_e|_{\textit{Specattrs}(e)} \in H(e)$. If $e = e_1 - e_2$ we make an analogous reasoning.
- If $e = e_1 \bowtie e_2$, then $\tau_{e_1} \sim \tau_{e_2}$, and $\tau_e = \tau_{e_1} \cup \tau_{e_2}$ by the type rule for join. Moreover, $\tau_{e_1}|_{\textit{Specattrs}(e_1)} \in H(e_1)$ and $\tau_{e_2}|_{\textit{Specattrs}(e_2)} \in H(e_2)$ by the induction hypothesis. Since $\tau_{e_1}|_{\textit{Specattrs}(e_1)} \subseteq \tau_e|_{\textit{Specattrs}(e)}$ and $\tau_{e_2}|_{\textit{Specattrs}(e_2)} \subseteq \tau_e|_{\textit{Specattrs}(e)}$, and since *Struc*$(e) =$ *Struc*$(e_1) \cup$ *Struc*(e_2), it follows that $\tau_e|_{\textit{Specattrs}(e)} \in H(e)$.

– If $e = \sigma_{\theta_i(A_1,\ldots,A_k)}(e')$, then $\tau_{e'}|_{\mathit{Specattrs}(e')} \in H(e')$ by the induction hypothesis. Furthermore, $(\tau_{e'}(A_1), \ldots, \tau_{e'}(A_k)) \in \varsigma(\theta_i)$ and $\tau_e = \tau_{e'}$ by the type rule for σ. For $1 \leq j \leq n$ we have

$$\theta_j(e) = \begin{cases} \theta_i(e') \cup \{(A_1, \ldots, A_k)\} & \text{if } i = j \\ \theta_j(e') & \text{otherwise.} \end{cases}$$

Hence, $\tau_e|_{\mathit{Specattrs}(e)} \in H(e)$. □

Using this lemma, we may conclude that typability is as least as difficult as non-uniform CSP.

Theorem 5. *If Θ is a finite, non-empty set of predicates and V is a subset of the relational algebra operators containing σ, then $\mathcal{H}(\mathit{Struc}(\Theta))$ is* LOGSPACE *reducible to $\mathcal{T}(\mathcal{R}_V^\Theta)$.*

Proof. We show that for every relational structure $\mathbf{A}$ we can create an expression $e \in \mathcal{R}_\sigma^\Theta$ (in logarithmic space) such that there is a homomorphism from $\mathbf{A}$ to $\mathit{Struc}(\Theta)$ if, and only if, there is a homomorphism from $\mathit{Struc}(e)$ to $\mathit{Struc}(\Theta)$. The result then follows by Lemma 4.

Let $\Theta = \{\theta_1, \ldots, \theta_n\}$ and let $\mathbf{A} = (C, R_1, \ldots, R_n)$ be a relational structure where the arity of R_i equals $|\theta_i|$. Let $\mathit{adom}(\mathbf{A})$ denote the *active domain* of $\mathbf{A}$, i.e., the set of all elements in C actually mentioned in one of the R_i. It is easy to see that there is a homomorphism from $\mathbf{A}$ to $\mathit{Struc}(\Theta)$ if, and only if, there is a homomorphism from $(\mathit{adom}(\mathbf{A}), R_1, \ldots, R_n)$ to $\mathit{Struc}(\Theta)$.

We now create e such that $\mathit{Struc}(e) = (\mathit{adom}(\mathbf{A}), R_1, \ldots, R_n)$. This is true whenever e is of the form $\sigma \ldots \sigma(r)$ such that for every R_i and every tuple $(c_1, \ldots, c_k)$ in R_i there is a subexpression of the form $\sigma_{\theta_i(c_1,\ldots,c_k)}(e')$ in e. Here we view $c_1, \ldots, c_k$ as attribute names. It is easy to see that we can create such an e in logarithmic space: we simply iterate over the tuples in $\mathbf{A}$, and in each iteration add an extra selection operator of the correct form to the expression built so far. □

Corollary 6. *Let V be a subset of the relational algebra operators containing σ. Then $\mathcal{T}(\mathcal{R}_V^\Theta)$ is* NP*-complete if $\mathcal{H}(\mathit{Struc}(\Theta))$ is.*

As we have noted before, there are structures $\mathbf{B}$ for which $\mathcal{H}(\mathbf{B})$ is NP-complete. For every such structure $\mathbf{B} = (C, S_1, \ldots, S_n)$ we can create a set of predicates Θ such that $\mathit{Struc}(\theta) = \mathbf{B}$. Indeed, we simply take Θ to contain predicates $\theta_1, \ldots, \theta_n$ such that $|\theta_i|$ equals the arity of S_i, and such that $\varsigma(\theta_i) = S_i$. Hence, there are predicate sets Θ for which $\mathcal{T}(\mathcal{R}_V^\Theta)$ is NP-complete.

On the positive side, the following corollary to Lemma 4 tells us that the complexity of $\mathcal{T}(\mathcal{R}_{\cup,-,\bowtie,\sigma}^\Theta)$ is in P when $\mathcal{H}(\mathit{Struc}(\Theta))$ is in P.

Corollary 7. *If Θ is a finite, non-empty set of predicates, then $\mathcal{T}(\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma})$ is* LOGSPACE *reducible to $\mathcal{H}(\mathit{Struc}(\Theta))$.*

This result cannot be generalized further to include π or ρ. To see why, let us fix a set of unary predicates $\Omega = \{\theta_1, \theta_2\}$ where $\varsigma(\theta_1) = \{0\}$ and $\varsigma(\theta_2) = \{1\}$. Here, 0 and 1 are base types. Note that such predicates will occur in practice. For instance, we can interpret θ_1 by "equals 5" with 0 being the base type `int` and θ_2 by "equals Mary" with 1 being the base type `string`.

Theorem 8. *With Ω the set of predicates described above, $\mathcal{H}(\mathit{Struc}(\Omega))$ is in* P*, but $\mathcal{T}(\mathcal{R}^{\Omega}_{\bowtie,\sigma,\pi})$ and $\mathcal{T}(\mathcal{R}^{\Omega}_{\bowtie,\sigma,\rho})$ are* NP*-complete.*

Proof. Obviously, $\mathbf{A} = (C, R_1, R_2) \in \mathcal{H}(\mathit{Struc}(\Omega))$ if, and only if, $R_1 \cap R_2 = \emptyset$, which can be checked in polynomial time.

We only need to show NP-hardness of $\mathcal{T}(\mathcal{R}^{\Omega}_{\cup,-,\bowtie,\sigma,\pi,\rho})$, for which we modify a reduction invented by Ohori and Buneman [4]. The reduction is from MONOTONE 3SAT [3]: decide whether there is a satisfying truth assignment for a given 3CNF boolean formula ϕ whose clauses are either all variables (called a positive clause) or all negated variables (called a negative clause).

Let $\phi = (a_1^1 \vee a_2^1 \vee a_3^1) \wedge \cdots \wedge (a_1^n \vee a_2^n \vee a_3^n)$ be such a formula. We will create an expression e_ϕ such that e_ϕ is typable if, and only if, ϕ is satisfiable. We will use the variables and negated variables $a_1^1, \ldots, a_3^n$ of ϕ as relation names. Intuitively, we encode truth assignments w on the set X of all variables in ϕ by type assignments Γ where $A \in \mathit{dom}(\Gamma(x))$ if, and only, if $w(x)$ is true and $A \in \mathit{dom}(\Gamma(\overline{x}))$ if, and only if, $w(x)$ is false. Here, we denote by x a variable and by $\overline{x}$ a negated variable.

Let us first define, for every variable x in ϕ, the expression:

$$e_x := \pi_B \sigma_{\theta_1(A)}(x \bowtie x_1) \bowtie \pi_B \sigma_{\theta_2(A)}(\overline{x} \bowtie x_2) \bowtie \pi_B \pi_{A,B}(x \bowtie \overline{x}).$$

Intuitively, e_x is used to verify that every type assignment under which e_ϕ is well-typed is indeed an encoding of a truth assignment. The whole expression is now defined by:

$$e_\phi := \bowtie_{x \in X} e_x \bowtie \bowtie_{i=1}^{n} \pi_B \pi_{A,B}(a_1^i \bowtie a_2^i \bowtie a_3^i).$$

It is clear that e_ϕ can be constructed from ϕ in logarithmic space.

Suppose that ϕ is satisfiable. Then there exists a satisfying truth assignment w on the variables of ϕ. To show that e_ϕ is typable, we construct the type assignment Γ on $\mathit{Relvars}(e)$ as follows. Let τ be a type which is undefined on all attributes except B. Let τ_1 and τ_2 be the types with domain

$\{A\}$ such that $\tau_1(A) = 0$ and $\tau_2(A) = 1$. If $w(x)$ is true, then we define

$$\begin{aligned} \Gamma(x) &:= \tau \cup \tau_1 & \Gamma(\overline{x}) &:= \emptyset \\ \Gamma(x_1) &:= \emptyset & \Gamma(x_2) &:= \tau \cup \tau_2 \end{aligned}$$

Otherwise, we define

$$\begin{aligned} \Gamma(x) &:= \emptyset & \Gamma(\overline{x}) &:= \tau \cup \tau_2 \\ \Gamma(x_1) &:= \tau \cup \tau_1 & \Gamma(x_2) &:= \emptyset \end{aligned}$$

The reader is asked to verify that e_x has output type τ under Γ. Since every clause in ϕ consists entirely of un-negated variables, or entirely of negated variables, and since by construction $\Gamma(x) \sim \Gamma(y)$ and $\Gamma(\overline{x}) \sim \Gamma(\overline{y})$ for every variable x and y, the subexpressions $(a_1^i \bowtie a_2^i \bowtie a_3^i)$ are well-typed under Γ. Moreover, since $w(a_1^i \vee a_2^i \vee a_3^i)$ is true, at least one of $\Gamma(a_1^i)$, $\Gamma(a_2^i)$ and $\Gamma(a_3^i)$ is defined on A and B. Since $\Gamma(a_1^i) \cup \Gamma(a_2^i) \cup \Gamma(a_3^i)$ is the type of $(a_1^i \bowtie a_2^i \bowtie a_3^i)$ under Γ, we know that $\pi_B\pi_{A,B}(a_1^i \bowtie a_2^i \bowtie a_3^i)$ also has type τ under Γ. Then e_ϕ is well-typed under Γ, since it is a join of subexpressions of type τ and since τ is certainly compatible with itself.

Conversely, suppose e_ϕ is typable. Then there exists a type assignment Γ on $\mathit{Relvars}(e)$ such that every subexpression of e is well-typed under Γ. In particular, e_x is well-typed under Γ for every variable x. Then Γ encodes a truth assignment. Indeed, the subexpression $\pi_B\pi_{A,B}(x \bowtie \overline{x})$ of e_x requires that $A \in \mathit{dom}(\Gamma(x))$ or $A \in \mathit{dom}(\Gamma(\overline{x}))$. However, if $A \in \mathit{dom}(\Gamma(x))$, then subexpression $\sigma_{\theta_1(A)}(x \bowtie x_1)$ of e_x requires $\Gamma(x)(A) = 0$, while subexpression $\sigma_{\theta_2(A)}(\overline{x} \bowtie x_2)$ requires $\Gamma(\overline{x})(A) = 1$ when $A \in \mathit{dom}(\Gamma(\overline{x}))$. Since subexpression $x \bowtie \overline{x}$ of e_x requires that $\Gamma(x)$ and $\Gamma(\overline{x})$ are compatible, we have $A \in \mathit{dom}(\Gamma(x))$ if, and only if, $A \notin \mathit{dom}(\Gamma(\overline{x}))$. Let w be the truth assignment such that $w(x)$ is true if, and only if, $A \in \mathit{dom}(\Gamma(x))$. Let $1 \leq i \leq n$. Since $\pi_{A,B}(a_1^i \bowtie a_2^i \bowtie a_3^i)$ is well-typed under Γ, the type of $a_1^i \bowtie a_2^i \bowtie a_3^i$ must defined on A. Hence, $\Gamma(a_j^i)$ is defined on A for some $1 \leq j \leq 3$. We have two cases. Either $a_j^i = x$ for some variable x, meaning that the i-th clause in ϕ is positive. Then $w(a_1^i \vee a_2^i \vee a_3^i)$ is true since $w(a_j^i)$ is true. Otherwise, $a_j^i = \overline{x}$ for some variable x and the i-th clause of ϕ is negative. Then $\Gamma(x)$ cannot be defined on A since $\Gamma(\overline{x})$ is defined on A. Hence, $w(x)$ is false which means $w(\overline{x})$ is true and thus $w(a_1^i \vee a_2^i \vee a_3^i)$ is true. Hence, ϕ is satisfiable.

To show NP-hardness of $\mathcal{T}(\mathcal{R}^{\Omega}_{\bowtie,\sigma,\rho})$ a similar reduction can be made: we define

$$\begin{aligned} e_x &:= \rho_{A/C_x}\sigma_{\theta_1(A)}(x \bowtie x_1) \bowtie \rho_{A/D_x}\sigma_{\theta_2(A)}(\overline{x} \bowtie x_2) \bowtie \rho_{A/E_x}(x \bowtie \overline{x}) \\ e_\phi &:= \bowtie_{x \in X} e_x \bowtie \bowtie_{i=1}^{n} \rho_{A/F_i}(a_1^i \bowtie a_2^i \bowtie a_3^i). \end{aligned}$$

Here the auxiliary attributes C_x, D_x, E_x, and F_i are used to prevent base-type clashes between the various subexpressions. $\square$

As a consequence, $\mathcal{T}(\mathcal{R}_V^\Omega)$ is NP-complete whenever V includes $\{\bowtie, \sigma, \pi\}$ or $\{\bowtie, \sigma, \rho\}$ as a subset of operators.

The predicate set Ω depends heavily on the presence of more than one base type. What is the complexity of deciding typability when we have only one base type, as in the original setting of Van den Bussche and Waller? As we will show, this can be done in polynomial time. This implies that we can at least efficiently check expressions for mistakes that require an attribute to be present and absent at the same time, as for example in $\rho_{A/B}(\pi_B(r))$.

Theorem 9. *Let Θ be a finite set of predicates over at most one base type, so $|\beta(\Theta)| = 1$. Then $\mathcal{T}(\mathcal{R}^\Theta_{\cup,-,\bowtie,\sigma,\pi,\rho})$ is in* P.

To prove this Theorem, we will show that we can always reformulate the typability problem as a non-uniform CSP which is solvable in polynomial time. Since $|\beta(\Theta)| = 1$, there can be no base-type clashes in an expression e, and hence we only need to verify that attributes are used consistently, i.e. that an attribute is not required to be present and absent at the same time.

We will record the requirements the type system makes on the presence or absence of attributes in an relational structure as follows. Let $e' \preceq e$ denote the fact that e' is a subexpression of e. To each expression $e \in \mathcal{R}^\Theta_{\cup,-,\bowtie,\sigma,\pi,\rho}$ we then associate the relational structure

$$\mathbf{A_e} = (C_e, D_e, U_e, E_e, J_e)$$

where

- C_e is a set of *variables* of the form $A^{e'}$ where e' is a subexpression of e and A is an attribute occurring in e;
- D_e is the set of variables $A^{e'}$ for which the type system requires that A is present in the output type of subexpression e' under any type assignment Γ which makes e well-typed:

$$\begin{aligned} D_e = &\{A^{e'} \mid \sigma_{\theta(B_1,\ldots,A,\ldots,B_n)}(e') \preceq e\} \\ &\cup \{A^{e'} \mid \pi_{B_1,\ldots,A,\ldots,B_n}(e') \preceq e\} \\ &\cup \{A^{e'}, B^{\rho_{A/B}(e')} \mid \rho_{A/B}(e') \preceq e\}; \end{aligned}$$

- U_e is the set of variables $A^{e'}$ for which the type system requires that A is absent in the output type of e' under any type assignment Γ which makes e well-typed:

$$\begin{aligned} U_e = &\{A^{\pi_{B_1,\ldots,B_n}(e')} \mid \pi_{B_1,\ldots,B_n}(e') \preceq e, A \notin \{B_1, \ldots, B_n\}\} \\ &\cup \{A^{\rho_{A/B}(e')}, B^{e'} \mid \rho_{A/B}(e') \preceq e\}; \end{aligned}$$

– E_e is the set of pairs of variables $(A^{e'}, B^{e''})$ for which the type system requires that A is present in the output type of e' under a type assignment which makes e well-defined, if, and only if, B is present in the output type of e'' under this assignment:

$$\begin{aligned} E_e = &\{(A^{e_1}, A^{e_2}), (A^{e_1 \cup e_2}, A^{e_1}) \mid e_1 \cup e_2 \preceq e, A \in \textit{Specattrs}(e)\} \\ &\cup \{(A^{e_1}, A^{e_2}), (A^{e_1 - e_2}, A^{e_1}) \mid e_1 - e_2 \preceq e, A \in \textit{Specattrs}(e)\} \\ &\cup \{(A^{\sigma_{\theta(A_1,\ldots,A_n)}(e')}, A^{e'}) \mid \sigma_{\theta(A_1,\ldots,A_n)}(e') \preceq e, A \in \textit{Specattrs}(e)\}; \end{aligned}$$

– J_e captures the relations between the attributes imposed by the type rule for join:

$$J_e = \{(A^{e_1 \bowtie e_2}, A^{e_1}, A^{e_2}) \mid e_1 \bowtie e_2 \preceq e, A \in \textit{Specattrs}(e)\}.$$

It is clear that e is typable if, and only if, the requirements made by the type system can be met, i.e., if there is a homomorphism from $\mathbf{A_e}$ to the structure $\mathbf{B} = (\{0,1\}, D, U, E, J)$ where

– D is used to verify that the attributes mentioned in D_e are actually present: $D = \{1\}$;
– U is used to verify that the attributes mentioned in U_e are actually absent: $U = \{0\}$;
– E is used to verify that for the pairs of variables $(A^{e'}, B^{e''})$ mentioned in E_e, A is present in the output type of e' if, and only if, B is present in the output type of e': $E = \{(0,0),(1,1)\}$; and
– J is used to verify that for the triples $(A^{e_1 \bowtie e_2}, A^{e_1}, A^{e_2})$ in J_e, the presence of A in the output type of $e_1 \bowtie e_2$ follows the type rule for join: $J = \{(1,1,1),(1,1,0),(1,0,1),(0,0,0)\}$.

Such a relational structure, where the domain contains only two elements, is called a *boolean structure*.

Lemma 10. *Let Θ be a finite set of predicates. If $|\beta(\Theta)| = 1$ and $e \in \mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma,\pi,\rho}$, then e is typable if, and only if, there is a homomorphism from $\mathbf{A_e}$ to $\mathbf{B}$.*

Proof. Suppose that e is well-typed under type assignment Γ. Then every subexpression e' of e is well-typed under Γ. Let $\tau_{e'}$ be the type of e' under Γ. Define the function f from C_e to $\{0,1\}$ such that $f(A^{e'}) = 1$ if, and only if, $A \in \textit{dom}(\tau_{e'})$. It is easy to show that f is a homomorphism from $\mathbf{A_e}$ to $\mathbf{B}$. For example, if $(A^{e_1 \cup e_2}, A^{e_1}) \in E_e$, then $e_1 \cup e_2 \preceq e$ by construction. By the type rule for union, we know that $\tau_{e_1} = \tau_{e_1 \cup e_2}$. Hence $f(A^{e_1}) = f(A^{e_1 \cup e_2})$ and thus $(f(A^{e_1 \cup e_2}), f(A^{e_1})) \in E$. Similar reasonings can be made for the other cases.

Conversely, let h be a homomorphism from $\mathbf{A_e}$ to $\mathbf{B}$. Let b be the single base type in $\beta(\Theta)$. For every subexpression e' of e we define $\tau_{e'}$ such that

$\tau_{e'}(A) = b$ if $f(A^{e'}) = 1$, and $\tau_{e'}$ is undefined on A otherwise. Let Γ be the type assignment such that $\Gamma(r) = \tau_r$ for every $r \in \mathit{Relvars}(e)$. It is easy to show by induction on e' that $\Gamma \vdash e' : \tau_{e'}$. For example, if $e' = \sigma_{\theta(A_1,\ldots,A_n)}(e'')$, then $\Gamma \vdash e'' : \tau_{e''}$ by the induction hypothesis. By construction, $A_i^{e''} \in D_e$ for $1 \leq i \leq n$. Hence, $f(A_i^{e''}) \in D = \{1\}$ and thus $A \in \mathit{dom}(\tau_{e''})$. Moreover, $(\tau_{e''}(A_1), \ldots, \tau_{e''}(A_n)) = (b, \ldots, b)$. Since $|\beta(\Theta)| = 1$, $\varsigma(\theta) = \{(b, \ldots, b)\}$, and hence $(\tau_{e''}(A_1), \ldots, \tau_{e''}(A_n)) \in \varsigma(\theta)$. By the type rule for selection, $\Gamma \vdash e' : \tau_{e''}$. Since $(A^{e'}, A^{e''}) \in E_e$, we know that $(f(A^{e'}), f(A^{e''})) \in E$. Hence, $f(A^{e'}) = f(A^{e''})$, $\tau_{e'} = \tau_{e''}$, and $\Gamma \vdash e' : \tau_{e'}$. Similar reasonings can be made for the other cases. □

We recall the following important theorem from constraint satisfaction theory, due to Shaefer [6]:

Theorem 11 (Shaefer's Dichotomy Theorem).

– *If* $\mathbf{B}$ *is a Boolean structure, then* $\mathcal{H}(\mathbf{B})$ *is in* P *or it is* NP*-complete.*
– *In particular, if every relation in a Boolean structure* $\mathbf{B}$ *is closed under the function* $g(x, y) = x \vee y$*, then* $\mathcal{H}(\mathbf{B})$ *is in* P.

An n-ary relation R is *closed* under the function $g(x, y)$ if for any two tuples $(a_1, \ldots, a_n)$ and $(b_1, \ldots, b_n)$ in R the tuple $(g(a_1, b_1), \ldots, g(a_n, b_n))$ is also in R. It is easy to see that every relation in $\mathbf{B}$ is closed under g. Theorem 9 then follows from this theorem and Lemma 10.

As a corollary to Theorem 9, $\mathcal{T}(\mathcal{R}^{\Theta'}_{\cup,-,\bowtie,\pi,\rho})$ is in P for any predicate set Θ', since $\mathcal{R}^{\Theta'}_{\cup,-,\bowtie,\pi,\rho} \subseteq \mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma,\pi,\rho}$ for all predicate sets Θ (and hence in particular for those with $|\beta(\Theta)| = 1$).

5 Conclusion

We have shown that deciding typability for the relational algebra is NP-complete for many settings. In particular, if V is a superset of $\{\times, \cup, \pi\}$, $\{\times, -, \pi\}$, $\{\times, -, \sigma\}$, $\{\times, \cup, \sigma\}$, or $\{\times, \rho\}$ then deciding typability for $\mathcal{R}^{\Theta}_{V}$ is NP-complete for any set of predicates Θ. Since every non-uniform constraint satisfaction problem can be expressed in $\mathcal{R}^{\Theta}_{V}$ when V includes σ, $\mathcal{T}(\mathcal{R}^{\Theta}_{V})$ is NP-complete whenever $\mathcal{H}(\mathit{Struc}(\Theta))$ is. Furthermore, there exists a very simple set of predicates Ω such that the constraint satisfaction problem for Ω is in P, but $\mathcal{T}(\mathcal{R}^{\Omega}_{V})$ is NP-complete for any superset V of $\{\bowtie, \pi, \sigma\}$ or $\{\bowtie, \sigma, \rho\}$. On the positive side, deciding typability of expressions in $\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma}$ is in P when the constraint satisfaction problem for Θ is in P. Furthermore, deciding typability of expressions in $\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma,\pi,\rho}$ where Θ mentions only one base type ($|\beta(\Theta)| = 1$) is also in P. This implies that we can efficiently check expressions in $\mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma,\pi,\rho}$ for errors requiring an attribute to be present and absent at the same time. Since

$\mathcal{R}^{\Theta'}_{\cup,-,\bowtie,\pi,\rho} \subseteq \mathcal{R}^{\Theta}_{\cup,-,\bowtie,\sigma,\pi,\rho}$, this also implies that $\mathcal{T}(\mathcal{R}^{\Theta'}_{\cup,-,\bowtie,\pi,\rho})$ is in P for any predicate set Θ'.

Acknowledgements. The author thanks Frank Neven, Jan Van den Bussche, Dirk Van Gucht and the anonymous referees for comments on an earlier draft of this paper.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. C. J. Date. *An Introduction to Database Systems*. Addison-Wesley, 6th edition edition, 1995.
3. M. R. Garey and D. S. Johnson. *Computer and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, 1979.
4. A. Ohori and P. Buneman. Type inference in a database programming language. In *Proceedings of the 1988 ACM conference on LISP and functional programming*. ACM Press, 1988.
5. B. C. Pierce. *Types and Programming Languages*. MIT Press, 2002.
6. T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the tenth annual ACM symposium on Theory of computing*, pages 216–226. ACM Press, 1978.
7. J. Van den Bussche and E. Waller. Polymorphic type inference for the relational algebra. *Jounal of Computer and System Sciences*, 64:694–718, 2002.