A Compositional Query Algebra for Second-Order Logic and Uncertain Databases

Christoph Koch
Department of Computer Science
Cornell University, Ithaca, NY, USA
koch@cs.cornell.edu

ABSTRACT

World-set algebra is a variable-free query language for uncertain databases. It constitutes the core of the query language implemented in MayBMS, an uncertain database system. This paper shows that world-set algebra captures exactly second-order logic over finite structures, or equivalently, the polynomial hierarchy. The proofs also imply that world-set algebra is closed under composition, a previously open problem.

1. INTRODUCTION

Developing suitable query languages for uncertain databases is a substantial research challenge that is only currently starting to get addressed. Conceptually, an uncertain database is a finite set of *possible worlds*, each one a relational database. *World-set algebra* (WSA) [4] is a query language for processing uncertain data in the spirit of relational algebra. WSA consists of the operations of relational algebra plus two further operations, one to introduce uncertainty and one to compute possible tuples across groups of possible worlds. WSA forms a core of the query language implemented in the probabilistic database management system MayBMS [13, 4, 3, 14, 12, 9]. The complexity and expressive power of world-set algebra have so far remained open.

The first main result of this paper is a proof that world-set algebra over uncertain databases precisely captures second-order logic (SO) over finite structures, or equivalently, the polynomial hierarchy. This seems to be a somewhat surprising coincidence, since the language was not designed with this result as a goal but by abstraction from a set of use cases from the contexts of hypothetical ("what-if") queries, decision support queries, and data cleaning. Viewed differently, WSA is a natural variable-free language equivalent to SO; it is to SO what relational algebra is to first-order logic. To the best of the author's knowledge, no other such language is known.

The fact that WSA exactly captures second-order logic is a strong argument to justify it as a query language for uncertain data. Second-order logic is a natural yardstick for languages for querying possible worlds. Indeed, second-order quantifiers are the essence of what-if reasoning in databases. World-set algebra seems to be a strong candidate for a core algebra for forming query plans and optimizing and executing them in uncertain database management systems.

It was left open in previous work whether world-set algebra is closed under composition, or whether definitions are adding to the expressive power of the language. Compositionality is a desirable and rather commonplace property of query algebras, but in the case of WSA it seems rather unlikely to hold. The reason for this is that the algebra contains an uncertainty-introduction operation that on the level of possible worlds is *nondeterministic*. First materializing a view and subsequently using it multiple times in the query is semantically quite different from composing the query with the view and thus obtaining several copies of the view definition that can independently make their nondeterministic choices. In the paper, evidence is given that seems to suggest that definitions are essential for the expressive power of WSA.

As the second main result, the paper nevertheless gives a proof that definitions do not add to the power of the language: WSA is indeed compositional. There is even a (nontrivial) practical linear-time translation from SO to WSA without definitions. This result, and the techniques for proving it, may also be relevant in other contexts. For example, it is shown that, essentially, self-joins can always be eliminated from classical relational algebra at the cost of introducing difference operators.

The proofs also imply that the data complexity of WSA is exactly characterized by the polynomial hierarchy. That is, each query of WSA is in one of the classes of the polynomial hierarchy and for each such class there is a WSA query that is complete for it. Moreover, WSA is PSPACE-complete with respect to combined complexity [19, 18].

For use as a query language for probabilistic databases, WSA has been extended very slightly by a tuple confidence computation operation (see e.g. [12]). The focus of this paper is on the nonprobabilistic language of [4]. For the efficient processing of queries of the probabilistic version of WSA, the confidence operation is naturally orthogonal to the remaining operations [3, 14, 12]. The expressiveness and complexity results obtained in the present paper constitute lower bounds for the probabilistic version of the language. But the non-probabilistic language is interesting and important in its own right: Many interesting queries can be phrased in terms of the alternatives possible in a data management scenario with uncertainty, without reference to the relative (probability) weights of these alternatives.

The structure of this paper is as follows. Section 2 establishes the connection between second-order logic and uncertain databases. Section 3 introduces world-set algebra and gives formal definitions of syntax and semantics. Section 4 proves that WSA exactly captures the expressive power of second-order logic over finite structures. These proofs assume the availability of a construct for making definitions (materializing views). Section 5 discusses the importance of being able to eliminate these definitions, and shows why it should seem rather surprising that definitions are not needed for capturing second-order logic. Section 6 finally proves that definitions can indeed be eliminated without loss of expressive power, and a construction for composition is given. We obtain from these results that WSA with or without definitions is complete for the polynomial hierarchy with respect to data complexity and PSPACE-complete with respect to combined complexity. We discuss related work in Section 8 and conclude in Section 9.

2. UNCERTAIN DATABASES

The schema of a relational database is a set of relation names together with a function *sch* that maps each relation name to a tuple of attribute names. The arity $|sch(R)|$ of a relation R is denoted by $ar(R)$. We use calligraphic symbols such as $\mathcal{A}$ and $\mathcal{B}$ for relational databases. Relations are sets of tuples (rather than multisets, as in SQL).

We will use the standard syntax of second-order logic (SO; see e.g. [15]). Its semantics is defined using the satisfaction relation $\models$, as usual. We use $R \subseteq S$ as a shortcut for $\forall \vec{x}\ R(\vec{x}) \Rightarrow S(\vec{x})$. Throughout this paper, we will only use second-order logic *relativized* to some given finite set of domain elements (say, D), as is common in finite model theory (cf. [15]). That is, first-order quantifiers $\exists x\ \phi$ are to be read as $\exists x\ D(x) \wedge \phi$ and second-order quantifiers $\exists R\ \phi$ are to be interpreted as $\exists R\ R \subseteq D^{ar(R)} \wedge \phi$.

An *uncertain database* over a given schema represents a finite set $W = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ of relational databases of that schema, called the *possible worlds*. One world among these is the true world, but we do not know which one.

We assume that the domain relation D is given in the input and is the same relation in all worlds. Specifically, it contains two symbols 0 and 1.

A *representation* for a finite set of possible worlds W over schema $(R_1, \ldots, R_k)$ is a pair of a relational database schema and a *representation formula* ω over that database schema with free second-order variables $R_1, \ldots, R_k$ and without free first-order variables such that ω is true on exactly those structures that are in W:

$$(R_1, \ldots, R_k) \models \omega \quad \Leftrightarrow \quad (R_1, \ldots, R_k) \in W.$$

EXAMPLE 2.1 (STANDARD REPRESENTATION). Consider a representation of an uncertain database by relations that associate with each tuple a local condition in the form of a conjunction of propositional literals. A possible world is identified by a truth assignment for the propositional variables used, and a tuple is in a possible world if the world's truth assignment makes the tuple's clause true.

A representation database consists of a set V of propositional variables, a relation L such that $L(c, p, 1)$ is true iff variable p occurs positively in conjunction c and $L(c, p, 0)$ is true iff variable p occurs negated in c, and a representation relation R_i' for each schema relation R_i which extends the schema of R_i by a column to associate each tuple with a conjunction.

Possible worlds are identified by subsets $P \subseteq V$ of variables that are true. A tuple $\vec{t}$ is in relation R_i in possible world P if $R_i'(\vec{t}, c)$ is true for some conjunction c and c is true for the variable assignment that makes the variables in P true and the other variables false.

The representation formula $\omega(R_1, \ldots, R_k)$ is

$$\exists P\ P \subseteq V \wedge \bigwedge_{i=1}^{k} \forall \vec{t}\ R_i(\vec{t}) \Leftrightarrow \exists c\ R_i'(\vec{t}, c) \wedge \\ \forall p\ (L(c, p, 0) \Rightarrow \neg P(p)) \wedge (L(c, p, 1) \Rightarrow P(p)).$$

This is the representation system that is essentially used in MystiQ [6], Trio [5], and MayBMS [3]. It is a special case of c-tables [10] in which variables are Boolean, local conditions are conjunctions of propositional literals, there is no global condition, and no variables occur in the data tuples themselves (just in the local conditions associated with the data tuples). Note that it is complete in the sense that it can represent any nonempty finite set of possible worlds. Moreover, it is succinct, i.e., the cardinality of the represented set of possible worlds is in general exponential in the size of the representation database. $\Box$

It is now easy to use second-order logic to express queries on uncertain databases encoded by a representation. For instance, Boolean query ϕ is possible if $\exists R_1 \cdots R_k\ \omega \wedge \phi$ and certain if $\forall R_1 \cdots R_k\ \omega \Rightarrow \phi$. Second-order logic allows us to use succinct representations of probabilistic databases, but also yields very powerful hypothetical queries that can ask questions about possible choices of *sets* of tuples. Such a choice of sets could be e.g. clusters of tuples in record matching (also known as deduplication and under many other names).

3. THE ALGEBRA

3.1 Syntax and Semantics

World-set algebra (WSA) consists of the operations of relational algebra (selection σ, projection π, renaming ρ, product $\times$, union $\cup$, and difference $-$), two additional operations repair-key$_{\vec{A}}$ and possible$_{\vec{A}}$, and definitions "let $R := Q$ in Q'" where R is a new relation symbol from a countably infinite supply of relation names; R may be used in Q'. Selection conditions are Boolean combinations of atomic inequalities over $<$, $\leq$, $=$, and $\neq$.[1] *WSA without definitions* is the set of WSA queries in which no let-expressions occur.

Each relation name occurring within a WSA expression can also be viewed as a relation *variable* in analogy with second-order logic. A relation variable is called *free* if it is not bound by a let-expression (as is R above). If a WSA query is evaluated against a probabilistic database, each of its free relation variables must occur in the schema of the database.

Conceptually all operations are evaluated in each possible world $\mathcal{A}$ individually.

[1]All results in this paper would still hold if selection conditions were restricted to atomic inequalities, but this footnote is too small to show this.

$$\begin{array}{rcl}
[\![\{\vec{t}\}]\!]_W^{\mathcal{A}} & := & \{\{\vec{t}\}\} \\
 & \ldots & \vec{t} \text{ constant tuple} \\
[\![R]\!]_W^{\mathcal{A}} & := & \{R^{\mathcal{A}}\} \\
[\![\theta(Q)]\!]_W^{\mathcal{A}} & := & \big\{\theta(R) \mid R \in [\![Q]\!]_W^{\mathcal{A}}\big\} \\
 & \ldots & \theta \in \{\sigma_\phi, \pi_{\vec{A}}, \rho_{A\to B}\} \\
[\![Q_1\ \theta\ Q_2]\!]_W^{\mathcal{A}} & := & \big\{R_1\ \theta\ R_2 \mid R_1 \in [\![Q_1]\!]_W^{\mathcal{A}},\ R_2 \in [\![Q_2]\!]_W^{\mathcal{A}}\big\} \\
 & \ldots & \theta \in \{\times, \cup, -\} \\
[\![\text{repair-key}_{\vec{A}}(Q)]\!]_W^{\mathcal{A}} & := & \big\{R' \mid R' \subseteq R \in [\![Q]\!]_W^{\mathcal{A}},\ \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R'),\ \vec{A} \text{ is a key for } R'\big\} \\
[\![\text{possible}_{\vec{A}}(Q)]\!]_W^{\mathcal{A}} & := & \big\{\bigcup\big\{R' \mid \mathcal{B} \in W,\ R' \in [\![Q]\!]_W^{\mathcal{B}},\ \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R')\big\} \mid R \in [\![Q]\!]_W^{\mathcal{A}}\big\} \\
[\![\text{let } R := Q \text{ in } Q']\!]_W^{\mathcal{A}} & := & \bigcup\big\{[\![Q']\!]^{(\mathcal{A},R)}_{\{(\mathcal{B},R') \mid \mathcal{B}\in W,\ R' \in [\![Q]\!]_W^{\mathcal{B}}\}} \mid R \in [\![Q]\!]_W^{\mathcal{A}}\big\}
\end{array}$$

Figure 1: Formal semantics of WSA.

- The operations of relational algebra are evaluated within $\mathcal{A}$ in the normal way.
- Given input relation R, repair-key$_{\vec{A}}(R)$ nondeterministically chooses a maximal repair of the functional dependency $\vec{A} \to sch(R)$ on R, that is, it returns a subset R' of R in which $\vec{A}$ is a (super)key such that there is no strict superset of R' which is a subset of R and in which $\vec{A}$ is a (super)key. Subset R' of R satisfies this requirement iff it satisfies the functional dependency and $\pi_{\vec{A}}(R') = \pi_{\vec{A}}(R)$.
- The operation possible$_{\vec{A}}(R)$ is the only operation that can look into possible worlds other than $\mathcal{A}$. It computes as result relation for $\mathcal{A}$ those tuples that occur in at least one of the relations R across the group of possible worlds that agree with $\mathcal{A}$ on $\pi_{\vec{A}}(R)$. In other words, a group is a maximal set of worlds in which the query $\pi_{\vec{A}}(R)$ computes the same relation.
- Definitions (statements "let $R := Q$ in Q'") temporarily extend $\mathcal{A}$ by a named relation R defined by query Q. Since the result of Q is nondeterministic in general, the overall set of possible worlds on which Q' runs (which is relevant for computing possible$_{\vec{A}}$) may increase.

As a convention, we use $\{\langle\rangle\}$ to represent truth and $\emptyset$ to represent falsity, over a nullary relation schema. We use expressions $\{0, 1\}$ and $\{r, g, b\}$ as shortcuts for $\{\langle 0\rangle\} \cup \{\langle 1\rangle\}$ and $\{\langle r\rangle\} \cup \{\langle g\rangle\} \cup \{\langle b\rangle\}$, respectively.

Example 3.1. In this example we will use an operation possible(Q) which computes the set of tuples that occur in the result of Q in at least one of the possible worlds. It will be shown in Section 3.2 that this operation is syntactic sugar and can be expressed in the base algebra defined above.

Given a relational database with two relations $V(V)$ and $E(\mathit{From}, \mathit{To})$ representing a graph (directed, or undirected if E is symmetric). Then the following WSA query Q returns true iff the graph is 3-colorable:

$$\text{let } R := \text{repair-key}_{sch(V)}\big(V \times \rho_C(\{r, g, b\})\big) \text{ in}$$
$$\text{possible}\big(\{\langle\rangle\} - \pi_\emptyset(\sigma_{1.V=2.\mathit{From} \wedge 2.\mathit{To}=3.V \wedge 1.C=3.C}(R \times E \times R))\big).$$

The possible relations R represent all the functions $V \to \{r, g, b\}$, and Q simply asks whether there is such an R such that there do not exist two adjacent nodes of the same color.

The corresponding SO sentence is

$$\exists R\ \phi_{R:V\to\{r,g,b\}} \wedge \neg\exists u, v, c\ R(u, c) \wedge E(u, v) \wedge R(v, c)$$

where $\phi_{R:V\to\{r,g,b\}}$ is a first-order sentence that states that R is a relation $\subset V \times \{r, g, b\}$ that satisfies the functional dependency $R : V \to \{r, g, b\}$ and for which $\pi_V(R) = V$. □

Formally, the semantics of world-set algebra is defined using a translation $[\![\cdot]\!]_W^{\mathcal{A}}$ given in Figure 1 such that for a context of a set of possible worlds W and a world $\mathcal{A} \in W$, R is a possible result of world-set algebra query Q iff $R \in [\![Q]\!]_W^{\mathcal{A}}$. If a query Q is run against an uncertain database W, then $[\![Q]\!]_W^{\mathcal{A}}$ gives the result of Q seen in possible world $\mathcal{A}$ of W.

3.2 Derived Operations: Syntactic Sugar

We will also consider the following operations, which are definable in the base language:

$$\begin{array}{rcl}
[\![\text{subset}(Q)]\!]_W^{\mathcal{A}} & := & \{R' \mid R' \subseteq R \in [\![Q]\!]_W^{\mathcal{A}}\} \\
[\![\text{choice-of}_{\vec{A}}(Q)]\!]_W^{\mathcal{A}} & := & \{\pi_{\vec{A}=\vec{a}}(R) \mid R \in [\![Q]\!]_W^{\mathcal{A}}, \vec{a} \in \pi_{\vec{A}}(R)\} \\
[\![\text{certain}_{\vec{A}}(Q)]\!]_W^{\mathcal{A}} & := & \big\{\bigcap\{R' \mid \mathcal{B} \in W, R' \in [\![Q]\!]_W^{\mathcal{B}}, \\
 & & \quad \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R')\} \\
 & & \quad \mid R \in [\![Q]\!]_W^{\mathcal{A}}\big\} \\
[\![\text{possible}(Q)]\!]_W^{\mathcal{A}} & := & \big\{\bigcup\{R \mid \mathcal{B} \in W, R \in [\![Q]\!]_W^{\mathcal{B}}\}\big\} \\
[\![\text{certain}(Q)]\!]_W^{\mathcal{A}} & := & \big\{\bigcap\{R \mid \mathcal{B} \in W, R \in [\![Q]\!]_W^{\mathcal{B}}\}\big\}
\end{array}$$

The operation subset nondeterministically chooses an arbitrary subset of its input relation. Conceptually, the operations subset and repair-key cause an exponential blowup of the possible worlds under consideration: For instance, on a certain database (i.e., consisting of a single possible world) subset(R) creates the powerset of relation R as the new set of possible worlds.[2]

[2]The subset operation is weaker than a true powerset operation such as in nested relational algebra with powerset [1]: that language has nonelementary complexity, while the complexity of WSA will be shown in this paper to be in PSPACE. The main difference is that in WSA, in each possible world, only one of the sets of the powerset are available,

The operation choice-of$_{\vec{A}}(R)$ nondeterministically chooses an $\vec{a} \in \pi_{\vec{A}}(R)$ and selects those tuples $\vec{t}$ of R for which $\vec{t}.\vec{A} = \vec{a}$.

The operation certain$_{\vec{A}}$ is the dual of possible$_{\vec{A}}$ and computes those tuples common to all the worlds that agree on $\pi_{\vec{A}}$.

The operations possible and certain compute the possible respectively certain tuples across *all* possible worlds. Using possible and certain, we can close the possible worlds semantics and ask for possible or certain tuples. For such queries $\mathcal{A}$ can be chosen arbitrarily (and the semantics function can be considered to be of the form $[\![Q]\!]_W$).

Proposition 3.2. *The operations choice-of$_{\vec{A}}$, certain$_{\vec{A}}$, and certain are definable in WSA with definitions. The operations subset and possible are expressible in WSA without definitions.*

Proof Sketch. The following equivalences relate the syntactic sugar to expressions of the base algebra.

$$\begin{aligned}
\text{choice-of}_{\vec{A}}(Q) &= \text{let } V := Q \text{ in} \\
&\quad \big(V \bowtie \text{repair-key}_{\emptyset}(\pi_{\vec{A}}(V))\big) \\
\text{certain}_{\vec{A}}(Q) &= \text{let } V := Q \text{ in} \\
&\quad \big(V - \text{possible}_{\vec{A}}\big(\text{possible}_{\vec{A}}(V) - V\big)\big) \\
\text{subset}(Q) &= \pi_{sch(Q)}(\sigma_{A=1}(\text{repair-key}_{sch(Q)}(\\
&\quad Q \times \rho_A(\{0,1\})))) \\
\text{possible}(Q) &= \pi_{sch(Q)}(\sigma_{B=1}(\text{possible}_A(\rho_{A,B}(\langle 1,1\rangle) \times Q \\
&\quad \cup\, \rho_{A,B,sch(Q)}(\langle 1,0,1,\ldots,1\rangle)))) \\
\text{certain}(Q) &= \pi_{sch(Q)}(\sigma_{B=1}(\text{certain}_A(\rho_{A,B}(\langle 1,1\rangle) \times Q \\
&\quad \cup\, \rho_{A,B,sch(Q)}(\langle 1,0,1,\ldots,1\rangle))))
\end{aligned}$$

Here, A and B are new column names not in *sch*(Q). The correctness of the proposition can be easily verified using the formal semantics definition given earlier in this section. □

Remark 3.3. In [4], it was shown that the fragment obtained from WSA by replacing repair-key by choice-of is a conservative extension of first-order logic. That is, every query of that language that maps from a single possible world to a single possible world is equivalent to a first-order query. It is not surprising that this is not true for full WSA.

3.3 Hypothetical Query Processing Example

The following example shows how WSA can be used for processing what-if queries.

Example 3.4. Consider the relational database of Figure 2(a) which represents employees working in companies and their skills. The query, a simplified decision support problem, will be stated in four steps.

1. *Suppose I choose* to buy exactly one company and, as a consequence, exactly one (key) employee leaves that company.

   $$\text{U} := \text{choice\_of}_{C,E}(\text{Company\_Emp})$$

   (This nondeterministically chooses a tuple from Company_Emp: a company to buy and an employee that will leave.)

2. Who are the remaining employees?

   $$\text{V} := \pi_{1.C,2.E}(U \bowtie_{1.C=2.C \wedge 1.E \neq 2.E} \text{Company\_Emp})$$

3. If I acquire that company, which skills can I obtain *for certain*?

   $$\text{W} := \text{certain}_C(\pi_{C,S}(V \bowtie \text{Emp\_Skills}))$$

   (This query computes the tuples of $V \bowtie$ Emp_Skills that are certain assuming that the company was chosen correctly – i.e., certain in the set of possible worlds that agree with this world on the C column.)

4. Now list the *possible* acquisition targets if the gain of the skill s_1 shall be guaranteed by the acquisition.

   $$\text{possible}(\pi_C(\sigma_{S=s_1}(W)))$$

Figure 2(b-d) shows the development of the uncertain database through steps 1 to 3. The first step creates five possible worlds corresponding to the five possible choices of company and leaving employee from relation Company_Emp. Steps two to four further process the query, and the overall result, which is the same in all five possible worlds, is

Result	C
	c_1

□

and the combination of information from several possible worlds proceeds by collecting only flat tuples and thus is quite restricted.

3.4 Remarks on the Formal Semantics

In this subsection, three facts are stated that formalize that the operations of WSA indeed behave as one would expect.

The first fact is that expressions of relational algebra can be evaluated in each possible world in parallel as claimed.

Lemma 3.5. *Let Q be a relational algebra expression. Then*

$$[\![Q]\!]^{\mathcal{A}}_{W} = [\![Q]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} = \{Q^{\mathcal{A}}\}$$

where $Q^{\mathcal{A}}$ denotes the standard semantics of relational algebra on a relational database $\mathcal{A}$.

The second fact concerns composition. Let $Q_1 \circ Q_2$ denote a WSA expression without definitions in which Q_2 occurs as a subexpression of Q_1. Here $Q_1 \circ Q_3$ would denote the expression obtained by replacing Q_2 in Q_1 by Q_3.

Lemma 3.6. $[\![Q_1 \circ Q_2]\!]^{\mathcal{A}}_{W} =$
$$\bigcup \Big\{ [\![Q_1 \circ R]\!]^{(\mathcal{A},R)}_{\{(\mathcal{B},R) \mid \mathcal{B} \in W, R \in [\![Q_2]\!]^{\mathcal{B}}_{W}\}} \mid R \in [\![Q_2]\!]^{\mathcal{A}}_{W} \Big\}.$$

The final fact is that definitions in subexpressions are unaffected by the operations (other than let-expressions) higher up in the expression tree and can be pulled to the top of the expression without modification:

Proposition 3.7. *For WSA queries Q, $\theta(Q_1, \ldots, Q_k)$, if view name V occurs only in Q_i,*

$$\theta(Q_1, \ldots, Q_{i-1}, (\textit{let } V := Q \textit{ in } Q_i), Q_{i+1}, \ldots, Q_k) = \big(\textit{let } V := Q \textit{ in } \theta(Q_1, \ldots, Q_k)\big).$$

Here $0 \le k \le 2$ and, for example, $k = 1$ for possible$_{\vec{A}}$.

Proof. The proof is by induction, showing that for any Q, $[\![Q]\!]^{(\mathcal{A},V)}_{W} = [\![Q]\!]^{\mathcal{A}}_{W'}$ where $W' = \{\mathcal{A} \mid (\mathcal{A}, V) \in W\}$ if relation

Company_Emp	C	E
	c_1	e_{11}
	c_1	e_{12}
	c_2	e_{21}
	c_2	e_{22}
	c_2	e_{23}

Emp_Skills	E	S
	e_{11}	s_1
	e_{12}	s_1
	e_{21}	s_1
	e_{21}	s_2
	e_{22}	s_2
	e_{23}	s_3

(a)

U_1	C	E
	c_1	e_{11}

U_2	C	E
	c_1	e_{12}

U_3	C	E
	c_2	e_{21}

U_4	C	E
	c_2	e_{22}

U_5	C	E
	c_2	e_{23}

(b)

V_1	C	E
	c_1	e_{12}

V_2	C	E
	c_1	e_{11}

V_3	C	E
	c_2	e_{22}
	c_2	e_{23}

V_4	C	E
	c_2	e_{21}
	c_2	e_{23}

V_5	C	E
	c_2	e_{21}
	c_2	e_{22}

(c)

W_1	C	S
	c_1	s_1

W_2	C	S
	c_1	s_1

W_3	C	S
	c_2	s_2

W_4	C	S
	c_2	s_2

W_5	C	S
	c_2	s_2

(d)

Figure 2: Database (a) and intermediate query results (b-d) of Example 3.4.

name V does not appear in Q. This is immediate for all operations other than possible$_{\vec{A}}$. Let $Q = \text{possible}_{\vec{A}}(Q')$ and let the induction hypothesis hold for Q', i.e., $\llbracket Q' \rrbracket_W^{(\mathcal{A},V)} = \llbracket Q' \rrbracket_{W'}^{\mathcal{A}}$. Then

$$\begin{aligned}
&\llbracket \text{possible}_{\vec{A}}(Q') \rrbracket_W^{(\mathcal{A},V)} = \\
&\quad \Big\{ \bigcup \{ R' \mid (\mathcal{B}, V') \in W, R' \in \llbracket Q' \rrbracket_W^{(\mathcal{B},V')}, \\
&\qquad \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R') \} \mid R \in \llbracket Q' \rrbracket_W^{(\mathcal{A},V)} \Big\} \\
&= \Big\{ \bigcup \{ R' \mid \mathcal{B} \in W', R' \in \llbracket Q' \rrbracket_{W'}^{\mathcal{B}}, \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R') \} \\
&\qquad \mid R \in \llbracket Q' \rrbracket_{W'}^{\mathcal{A}} \Big\} \\
&= \llbracket \text{possible}_{\vec{A}}(Q') \rrbracket_{W'}^{\mathcal{A}}.
\end{aligned}$$

Now we apply the fact just proven to the subqueries Q_j for $j \neq i$. By definition,

$$\begin{aligned}
\llbracket \text{let } V := Q \text{ in } \theta(Q_1, \dots, Q_k) \rrbracket_{W'}^{\mathcal{A}} &= \\
\{ \llbracket \theta(Q_1, \dots, Q_k) \rrbracket_W^{(\mathcal{A},V)} &\mid V \in \llbracket Q \rrbracket_{W'}^{\mathcal{A}} \}.
\end{aligned}$$

We distinguish between the various operations θ. For relational algebra,

$$\begin{aligned}
&\llbracket \theta(Q_1, \dots, Q_k) \rrbracket_W^{(\mathcal{A},V)} = \\
&\quad \Big\{ \theta(R_1, \dots, R_k) \mid \bigwedge_j R_j \in \llbracket Q_j \rrbracket_W^{(\mathcal{A},V)} \Big\} \\
&= \Big\{ \theta(R_1, \dots, R_k) \mid R_i \in \llbracket Q_i \rrbracket_W^{(\mathcal{A},V)}, \bigwedge_{j \neq i} R_j \in \llbracket Q_j \rrbracket_{W'}^{\mathcal{A}} \Big\}
\end{aligned}$$

because V only occurs in Q_i and $\llbracket Q_j \rrbracket_W^{(\mathcal{A},V)} = \llbracket Q_j \rrbracket_{W'}^{\mathcal{A}}$ for $j \neq i$. Thus

$$\begin{aligned}
&\llbracket \text{let } V := Q \text{ in } \theta(Q_1, \dots, Q_k) \rrbracket_{W'}^{\mathcal{A}} = \\
&\quad \Big\{ \theta(R_1, \dots, R_k) \mid \underbrace{R_i \in \llbracket Q_i \rrbracket_W^{(\mathcal{A},V)}, V \in \llbracket Q \rrbracket_{W'}^{\mathcal{A}}}_{R_i \in \llbracket \text{let } V := Q \text{ in } Q_i \rrbracket_{W'}^{\mathcal{A}}}, \\
&\qquad \bigwedge_{j \neq i} R_j \in \llbracket Q_j \rrbracket_{W'}^{\mathcal{A}} \Big\} \\
&= \llbracket \theta(Q_1, \dots, Q_{i-1}, (\text{let } V := Q \text{ in } Q_i), Q_{i+1}, \dots, Q_k) \rrbracket_{W'}^{\mathcal{A}}
\end{aligned}$$

The proof for the remaining operations proceeds similarly. □

In other words, definitions can be considered "global". Without loss of generality we could assume that each WSA query is of the form

$$\text{let } V_1 := Q_1 \text{ in } (\cdots (\text{let } V_k := Q_k \text{ in } Q) \cdots)$$

where Q does not contain definitions.

Observe that in the case of binary relational algebra operations θ, the set of possible worlds $\llbracket Q_1 \; \theta \; Q_2 \rrbracket_W^{\mathcal{A}}$ is obtained by pairing relations in the results of $\llbracket Q_1 \rrbracket_W^{\mathcal{A}}$ and $\llbracket Q_2 \rrbracket_W^{\mathcal{A}}$. This is consistent with the intuition that θ is applied to possible worlds $\mathcal{B}$ that contain two relations $R_1^{\mathcal{B}}$ and $R_2^{\mathcal{B}}$ and the result in $\mathcal{B}$ is $R_1^{\mathcal{B}} \; \theta \; R_2^{\mathcal{B}}$: Proposition 3.7 implies that

$$\begin{aligned}
\theta(Q_1, \dots, Q_k) \;=\; \big(&\text{let } V_1 := Q_1 \text{ in } (\cdots \\
&(\text{let } V_k := Q_k \text{ in } \theta(V_1, \dots, V_k)) \cdots)\big).
\end{aligned}$$

4. WSA WITH DEFINITIONS CAPTURES SO LOGIC

In this section, it is shown that WSA with definitions has exactly the same expressive power as second-order logic over finite structures.

We must first make precise how second-order logic will be compared to WSA, since second-order logic queries are usually not "run" on uncertain databases. We will consider WSA queries that are evaluated against a (single-world) relational database $\mathcal{A}$ representing an uncertain database (e.g., using the standard representation of Example 2.1). We already know that arbitrary uncertain databases (that is, nonempty finite sets of possible worlds) can be so represented: This assumption means no loss of generality. A WSA query Q constructs the uncertain database from the representation and is always evaluated as $[\![Q]\!]^{\mathcal{A}}_{\{\mathcal{A}\}}$, precisely as sketched at the end of Section 2. For our expressiveness comparisons, we will focus on WSA expressions that close the possible worlds semantics and return a singleton result: Only these have strictly a correspondence with second-order logic. However, in some cases, when it is meaningful to generalize from this, we will do so.

We will study second-order formulae which may have free first-order variables but in which the second-order variables are bound to input relations. We will generalize from this, proving a more powerful result, in Section 6.

By equivalence between a WSA query Q and an SO formula ϕ, we thus mean that

$$[\![Q]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} = \{\{\vec{a} \in D^{ar(Q)} \mid \mathcal{A} \vDash \phi[\vec{a}]\}\}$$

for all relational databases $\mathcal{A}$ of suitable schema.

Theorem 4.1. *For every SO query, there is an equivalent WSA query with definitions.*

Proof. We may assume without loss of generality that the SO query is a first-order query (which of course may use first-order quantification) prefixed by a sequence of second-order quantifiers. The theorem is shown by induction.

Induction start: FO queries can be translated to relational algebra by a well-known translation known in the database context as one direction of Codd's Theorem (cf. [2]).

Induction step $\exists R_{k+1}(\subseteq D^l)\ \phi$ (second-order existential quantification): Let ϕ be an SO formula with free second-order variables $R_1, \dots, R_{k+1}$ and free first-order variables $\vec{x}$ where R_{k+1} has arity l. Let Q_ϕ be a WSA expression equivalent to ϕ given by the induction hypothesis. Without loss of generality, we may assume that the relations $R_1, \dots, R_k, Q_\phi$ have disjoint schemas. Let

$$Q := (\text{let } R_{k+1} := \text{subset}(D^l) \text{ in } \pi_{sch(Q_\phi)}(Q'))$$

where D is the domain relation,

$$Q' = \text{possible}_{sch(1_{R_1})\dots sch(1_{R_k})}(1_{R_1} \times \dots \times 1_{R_k} \times Q_\phi),$$

and

$$1_{R_i} = R_i \times \{\langle 1\rangle\} \cup (D^{ar(R_i)} - R_i) \times \{\langle 0\rangle\}.$$

(Note that the relations 1_{R_i} will play a prominent role in later parts of this paper.) We prove that

$$(R_1, \dots, R_k, \vec{x}) \vDash \exists R_{k+1}(\subseteq D^l)\ \phi \quad \Leftrightarrow \quad \vec{x} \in R_Q$$

where $\{R_Q\} = [\![Q]\!]^{(R_1,\dots,R_k)}_W$. By definition of $[\![\cdot]\!]$,

$$[\![Q]\!]^{(R_1,\dots,R_k)}_W = \{\pi_{sch(Q_\phi)}([\![Q']\!]^{(R_1,\dots,R_{k+1})}_{W'}) \mid R_{k+1} \subseteq D^l\}$$

where

$$W' = \{(R_1, \dots, R_{k+1}) \mid (R_1, \dots, R_k) \in W, R_{k+1} \subseteq D^l\}.$$

We may assume a nonempty domain D, so the result of $1_{R_1} \times \dots \times 1_{R_k}$ is never empty. The mapping $(R_1, \dots, R_k) \mapsto 1_{R_1} \times \dots \times 1_{R_k}$ is injective. Query Q will therefore group the possible outcomes of Q_ϕ for the various choices of R_{k+1} by $R_1, \dots, R_k$.

Formally, by definition of $[\![\cdot]\!]$,

$$\begin{aligned}
&[\![Q']\!]^{(R_1,\dots,R_{k+1})}_{W'} = \\
&\quad \Big\{ \bigcup \{ 1_{R_1} \times \dots \times 1_{R_k} \times [\![Q_\phi]\!]^{(R_1,\dots,R_k,R'_{k+1})}_{W'} \mid \\
&\qquad (R_1, \dots, R_k, R'_{k+1}) \in W' \} \mid (R_1, \dots, R_{k+1}) \in W' \Big\} \\
&= \Big\{ 1_{R_1} \times \dots \times 1_{R_k} \times \\
&\qquad \bigcup \{ [\![Q_\phi]\!]^{(R_1,\dots,R_k,R'_{k+1})}_{W'} \mid R'_{k+1} \subseteq D^l \} \Big\}.
\end{aligned}$$

Thus, in a given world $(R_1, \dots, R_k)$, Q produces exactly one world as the result,

$$\begin{aligned}
[\![Q]\!]^{(R_1,\dots,R_k)}_W &= \Big\{ \bigcup \{ [\![Q_\phi]\!]^{(R_1,\dots,R_k,R'_{k+1})}_{W'} \mid R'_{k+1} \subseteq D^l \} \Big\} \\
&= \{R_Q\}
\end{aligned}$$

and this captures exactly second-order existential quantification.

The WSA expression for universal second-order quantifiers $\forall R_{k+1}(\subseteq D^l)\ \phi$ is similar. Alternatively, $\forall R_{k+1}\ \phi$ can also be taken as $\neg\exists R_{k+1}\ \neg\phi$, where complementation with respect to D is straightforward using the difference operation. □

Example 4.2. Σ_2-QBF is the following Σ_2^P-complete decision problem. Given two disjoint sets of propositional variables V_1 and V_2 and a DNF formula ϕ over the variables of V_1 and V_2, does there exist a truth assignment for the variables V_1 such that ϕ is true for all truth assignments for the variables V_2?

Instances of this problem shall be represented by sets V_1 and V_2, a set C of ids of clauses in ϕ, and a ternary relation $L(C, P, S)$ such that $\langle c, p, 1\rangle \in L$ (resp., $\langle c, p, 0\rangle \in L$) iff propositional variable p occurs positively (resp., negatively) in clause c of ϕ, i.e.,

$$\phi = \bigvee_{c \in C} \bigwedge_{\langle c,p,1\rangle \in L} p \wedge \bigwedge_{\langle c,p,0\rangle \in L} \neg p.$$

The QBF is true iff second-order sentence

$$\exists P_1\ (P_1 \subseteq V_1) \wedge \forall P_2\ (P_2 \subseteq V_2) \Rightarrow \psi$$

is true, where ψ is the first-order sentence

$$\begin{aligned}
&\exists c\ \neg\exists p\ \big(L(c, p, 0) \wedge (P_1(p) \vee P_2(p))\big) \vee \\
&\qquad\qquad \big(L(c, p, 1) \wedge \neg(P_1(p) \vee P_2(p))\big).
\end{aligned}$$

which asserts the truth of ϕ: that there is a clause c in ϕ of which no literal is inconsistent with the truth assignment $p \mapsto (p \in P_1 \cup P_2)$. By Theorem 4.1, this can be expressed as the Boolean WSA query

$$\begin{aligned}
&\text{let } P_1 := \text{subset}(V_1) \text{ in possible}\big(\{\langle\rangle\} \\
&- \text{let } P_2 := \text{subset}(V_2) \text{ in possible}_{sch(P_1)}(1_{P_1} \times (\{\langle\rangle\} - Q))\big)
\end{aligned}$$

where

$$\begin{aligned}
Q = \pi_\emptyset\big(C - \pi_C\big(&(\sigma_{S=0}(L) \bowtie (P_1 \cup P_2)) \cup \\
&(\sigma_{S=1}(L) \bowtie ((V_1 \cup V_2) - (P_1 \cup P_2)))\big)\big)
\end{aligned}$$

is relational algebra for ψ. □

We now prove the converse direction of the expressiveness result.

Theorem 4.3. *For every WSA query, there is an equivalent second-order logic query.*

Proof Sketch. The proof revolves around the definition of a function $[\![\cdot]\!]_{so}$ that maps each WSA expression Q to an SO formula $[\![Q]\!]_{so}$ with free second-order variables $\vec{R}$ and R_Q (the "query result") and without free first-order variables such that $[\![Q]\!]_{so}$ and Q are equivalent in the sense that $[\![Q]\!]_{so}$ is true on structure $(\mathcal{A}, \vec{R}, R_Q)$ iff R_Q is among the possible results of Q starting from possible world $(\mathcal{A}, \vec{R})$. We can state this notion of correctness, which is the hypothesis of the following induction along the structure of the WSA expression, formally as

$$(\mathcal{A}, \vec{R}, R_Q) \vDash [\![Q]\!]_{so} \quad \Leftrightarrow \quad R_Q \in [\![Q]\!]_W^{(\mathcal{A},\vec{R})}$$

for

$$W = \Big\{(\mathcal{A}, \vec{R}) \mid (\mathcal{A}, \vec{R}) \vDash \bigwedge_{V \text{ in } \vec{R}} \psi_V\Big\}.$$

For a WSA expression Q that closes the possible world semantics (i.e., $[\![Q]\!]_{\{\mathcal{A}\}}^{\mathcal{A}} = \{R_Q\}$ is a singleton set), we can define an SO formula with free first-order variables but without free second-order variables that computes R_Q as $\exists R_Q\ [\![Q]\!]_{so} \wedge R_Q(x_1, \dots, x_{ar(Q)})$.

Here the free second-order variables $\vec{R}$ are also the names of the views defined (using let-expressions) along the path from the root of the parse tree of the query to the subexpression Q. A formula ψ_V is identified by the name of the view relation V, assuming without loss of generality that each view name is introduced only once by a let expression across the entire query. The formulae ψ_V will be defined below.

For the operations θ of relational algebra,

$$\begin{aligned}
&[\![\theta(Q_1, \dots, Q_{ar(\theta)})]\!]_{so}(\vec{R}, R_Q) \ := \\
&\qquad \exists R_{Q_1} \cdots R_{Q_{ar(\theta)}} \ \Big(\bigwedge_{i=1}^{ar(\theta)} [\![Q_i]\!]_{so}(\vec{R}, R_{Q_i})\Big) \\
&\qquad\qquad \wedge\ \forall \vec{x}\ R_Q(\vec{x}) \Leftrightarrow \phi_{\theta(Q_1,\dots,Q_{ar(\theta)})}(\vec{x})
\end{aligned}$$

where $0 \le ar(\theta) \le 2$, $\phi_S(\vec{x}) := S(\vec{x})$, S is either a relation from $\mathcal{A}$ or a second-order variable from $\vec{R}$, and

$$\begin{aligned}
\phi_{\{\vec{t}\}}(\vec{x}) &:= \vec{x} = \vec{t} \\
\phi_{Q_1 \cup Q_2}(\vec{x}) &:= R_{Q_1}(\vec{x}) \vee R_{Q_2}(\vec{x}) \\
\phi_{Q_1 - Q_2}(\vec{x}) &:= R_{Q_1}(\vec{x}) \wedge \neg R_{Q_2}(\vec{x}) \\
\phi_{Q_1 \times Q_2}(\vec{x}, \vec{y}) &:= R_{Q_1}(\vec{x}) \wedge R_{Q_2}(\vec{y}) \\
\phi_{\sigma_\gamma(Q)}(\vec{x}) &:= R_Q(\vec{x}) \wedge \gamma \\
\phi_{\pi_{\vec{x}}(Q)}(\vec{x}) &:= \exists \vec{y}\ R_Q(\vec{x}, \vec{y}) \\
\phi_{\rho_{\vec{x}\to\vec{y}}(Q)}(\vec{y}) &:= \exists \vec{x}\ R_Q(\vec{x}) \wedge \vec{x} = \vec{y}.
\end{aligned}$$

It is easy to verify that for any tuple $\vec{x}$ and relational algebra operation θ, $(\mathcal{A}, R_{Q_1}, \dots, R_{Q_{ar(\theta)}}) \vDash \phi_{\theta(Q_1,\dots,Q_{ar(\theta)})}(\vec{x})$ if and only if $\vec{x}$ is a result tuple of relational algebra query $\theta(R_{Q_1}, \dots, R_{Q_{ar(\theta)}})$. Assume that the induction hypothesis holds for the subqueries $Q_1, \dots, Q_{ar(\theta)}$, i.e., $(\mathcal{A}, \vec{R}, R_{Q_i}) \vDash [\![Q_i]\!]_{so}$ if and only if $R_{Q_i} \in [\![Q_i]\!]_W^{(\mathcal{A},\vec{R})}$ for $1 \le i \le ar(\theta)$. The formula $[\![\theta(Q_1, \dots, Q_{ar(\theta)})]\!]_{so}$ just states that R_Q is a relation consisting of exactly those tuples $\vec{x}$ that satisfy $\phi_{\theta(Q_1,\dots,Q_{ar(\theta)})}(\vec{x})$ for a choice of possible results $R_{Q_i} \in [\![Q_i]\!]_W^{(\mathcal{A},\vec{R})}$ of the subqueries Q_i, for $1 \le i \le ar(\theta)$. But this is exactly the definition of $[\![\theta(Q_1, \dots, Q_{ar(\theta)})]\!]_W^{(\mathcal{A},\vec{R})}$.

This in particular covers the nullary operations of relational algebra ($\{\vec{t}\}$ and R), which form the induction start.

The remaining operations are those special to WSA (with definitions) and can be defined as shown in Figure 3. Here "$\vec{A}$ is a key for R" and $\pi_{\vec{A}}(R) = \pi_{\vec{A}}(R')$ are easily expressible in FO.

It is straightforward to verify the correctness of $[\![\cdot]\!]_{so}$ for subset and repair-key: The definitions of $[\![\cdot]\!]_{so}$ and $[\![\cdot]\!]$ essentially coincide.

Similarly, the correctness of the definition of $[\![\cdot]\!]_{so}$ for let is easy to verify. Here we also define the formulae ψ_V.

Finally, $[\![\text{possible}_{\vec{A}}(Q_1)]\!]_{so}$ makes reference to world-set W and for that purpose uses the formulae ψ_V: Indeed, the worlds in W are exactly those structures that satisfy all the ψ_V for relations V defined by let expressions on the path from the root of the query to the current subexpression $\text{possible}_{\vec{A}}(Q_1)$. The definition $[\![\text{possible}_{\vec{A}}(Q_1)]\!]_{so}$ is again very close to the definition of $[\![\text{possible}_{\vec{A}}(Q_1)]\!]$, and its correctness is straightforward to verify. □

Note that by eliminating the definitions ψ_V in the proof of Theorem 4.3 we in general obtain an exponential-size formula. However, the proof construction translates WSA *without definitions* to SO in polynomial time.

5. WHY WE ARE NOT DONE

The proof that WSA with definitions can express any SO query may seem to settle the expressiveness question for our language. However, understanding WSA without definitions is also important, for two reasons. First, it is a commonplace and desirable property of query algebras that they be compositional, i.e., that the power to define views is not needed for the expressive power, and all views can be eliminated by composing the query. Second, if this property does not hold, it means that in general we have to precompute and materialize views. And indeed, superficially we would expect that WSA is not compositional in that respect: it supports nondeterministic operations (repair-key and/or subset). If a view definition V contains such a nondeterministic operation and a query uses V at least twice, replacing each occurrence with the definition will not be equivalent because the two copies of the definition of V will produce different relations in some worlds. For example, (let $V :=$ subset(U) in $V \bowtie V$) is not at all equivalent to subset$(U) \bowtie$ subset(U).

The question remains whether for each WSA query there is an equivalent query in WSA without definitions via a less direct rewriting. The answer to this question is less obvious. Our language definition has assumed repair-key to be the base operation and subset definable using WSA with repair-key. Indeed, in WSA with definitions, either one can be defined using the other. However, it can be shown that repair-key cannot be expressed using subset without using definitions even though subset can guess subsets and appears comparable in expressiveness to repair-key.

Consider possible worlds databases in which each relation

$$
\begin{array}{rcl}
[\![\text{subset}(Q_1)]\!]_{so}(\vec{R}, R_Q) & := & \exists R_{Q_1}\ [\![Q_1]\!]_{so}(\vec{R}, R_{Q_1}) \wedge R_Q \subseteq R_{Q_1} \\
[\![\text{repair-key}_{\vec{A}}(Q_1)]\!]_{so}(\vec{R}, R_Q) & := & \exists R_{Q_1}\ [\![Q_1]\!]_{so}(\vec{R}, R_{Q_1}) \wedge R_Q \subseteq R_{Q_1} \\
& \wedge & \vec{A} \text{ is a key for } R_Q \\
& \wedge & \neg\exists R'_Q\ R_Q \subset R'_Q \subseteq R_{Q_1} \wedge \vec{A} \text{ is a key for } R'_Q \\
[\![\text{let } V := Q_1 \text{ in } Q_2]\!]_{so}(\vec{R}, R_Q) & := & \exists V\ \psi_V \wedge [\![Q_2]\!]_{so}(\vec{R}, V, R_Q) \\
& & \text{and define } \psi_V := [\![Q_1]\!]_{so}(\vec{R}, V) \\
[\![\text{possible}_{\vec{A}}(Q_1)]\!]_{so}(\vec{R}, R_Q) & := & \exists R_{Q_1}\ [\![Q_1]\!]_{so}(\vec{R}, R_{Q_1}) \wedge \forall \vec{x}\ R_Q(\vec{x}) \Leftrightarrow \\
& & \exists \vec{R}\ \Big(\Big(\bigwedge_{V \text{ in } \vec{R}} \psi_V \Big) \wedge \exists R'_{Q_1}\ [\![Q_1]\!]_{so}(\vec{R}, R'_{Q_1}) \\
& & \quad \wedge\ \pi_A(R_{Q_1}) = \pi_A(R'_{Q_1}) \ \wedge\ R'_{Q_1}(\vec{x}) \Big)
\end{array}
$$

Figure 3: Definition of $[\![\cdot]\!]_{so}$ for operations not part of relational algebra.

$$
\begin{array}{rcl}
[\![\theta]\!]_{ndef}(W_1, \ldots, W_{ar(\theta)}) & := & \{\theta(R_1, \ldots, R_{ar(\theta)}) \mid R_1 \in W_1, \ldots, R_{ar(\theta)} \in W_{ar(\theta)}\} \\
& & \ldots \text{ where } \theta \text{ is an operation of relational algebra} \\
[\![\text{repair-key}_{\vec{A}}]\!]_{ndef}(W) & := & \{R \mid R \subseteq R' \in W, \pi_A(R) = \pi_A(R'), \vec{A} \text{ is a key for } R\} \\
[\![\text{subset}]\!]_{ndef}(W) & := & \{R \mid R \subseteq R' \in W\} \\
[\![\text{possible}_{\vec{A}}]\!]_{ndef}(W) & := & \Big\{ \bigcup\{R' \in W \mid \pi_{\vec{A}}(R) = \pi_{\vec{A}}(R')\} \mid R \in W \Big\}
\end{array}
$$

Figure 4: Definition of $[\![\cdot]\!]_{ndef}$.

is independent from the other relations, i.e., the world set is of the form

$$\{(R_1, \ldots, R_k) \mid R_1 \in W_1, \ldots, R_k \in W_k\}.$$

WSA without definitions on such *relation-independent databases* gives rise to a much simpler and more intuitive semantics definition than the one of Section 3, via the function $[\![\cdot]\!]_{ndef}$ defined in Figure 4.

The correctness of this alternative semantics definition, stated next, is easy to verify.

PROPOSITION 5.1. *For relation-independent databases and WSA without definitions, $[\![\cdot]\!]_{ndef}$ is equivalent to $[\![\cdot]\!]$ in the sense that for any operation θ,*

$$\{[\![\theta(Q_1, \ldots, Q_{ar(\theta)})]\!]^{\mathcal{A}}_W \mid \mathcal{A} \in W\} = [\![\theta]\!]_{ndef}(W_1, \ldots, W_{ar(\theta)})$$

where $W_i = \bigcup\{[\![Q_i]\!]^{\mathcal{A}}_W \mid \mathcal{A} \in W\}$ for all $1 \leq i \leq ar(\theta)$.

The following result asserts that adding subset to relational algebra yields little expressive power. By the existence of a supremum of a set of worlds W, we assert the existence of an element $(\bigcup W) \in W$, denoted $\sup(W)$. An infimum is a set $\inf(W) := (\bigcap W) \in W$.

THEOREM 5.2. *On relation-independent databases, any world-set computable using relational algebra extended by the operation subset has a supremum and an infimum.*

Proof. The nullary relational algebra expressions ($\{\vec{t}\}$ and R) yield just a singleton world-set, and the single world is both the supremum and the infimum. Given a world-set W, $\sup([\![\text{subset}]\!]_{ndef}(W)) := \sup(W)$, $\inf([\![\text{subset}]\!]_{ndef}(W)) := \emptyset$. For a positive relational algebra expression θ, $\sup([\![\theta]\!]_{ndef}(W_1, \ldots, W_k)) := \theta(\sup(W_1), \ldots, \sup(W_k))$ and $\inf([\![\theta]\!]_{ndef}(W_1, \ldots, W_k)) := \theta(\inf(W_1), \ldots, \inf(W_k))$. For relational difference, it can be verified that $\sup([\![-]\!]_{ndef}(W_1, W_2)) := \sup(W_1) - \inf(W_2)$ and $\inf([\![-]\!]_{ndef}(W_1, W_2)) := \inf(W_1) - \sup(W_2)$. It is easy to verify the correctness of these definitions, and together they yield the theorem. $\Box$

Thus, not even $\text{repair-key}_\emptyset(\{0, 1\}) = \big\{\{0\}, \{1\}\big\}$ can be defined.

COROLLARY 5.3. *The set of worlds $\big\{\{0\}, \{1\}\big\}$ is not definable in relational algebra extended by subset.*

In contrast, $\text{repair-key}_{sch(U)}(U \times \{0, 1\})$ can be defined as follows in the language fragment of relational algebra plus subset if definitions are available:

$$\text{let } R := \text{subset}(U) \text{ in } (R \times \{\langle 1\rangle\} \cup (U - R) \times \{\langle 0\rangle\}).$$

Thus, removing definitions seems to cause a substantial reduction of expressive power. In the remainder of this paper, we study whether $\text{possible}_{\vec{A}}$ and repair-key can offset this.

Before we move on, another simple result shall be stated that gives an intuition for the apparent weakness of WSA without definitions. If a view is defined by a query that involves one of the nondeterministic operations ($\text{possible}_{\vec{A}}$ or repair-key), then this view can only be used at one place in the query if the query is to be composed with the view. However, subsequent relational algebra operations will be *monotonic* with respect to that view.

To be precise, by monotonicity of a query Q in one input relation R, we will, throughout this paper, mean that Q satisfies one of the following two properties:

1. For all instances $\mathcal{A}, \mathcal{B}$, if $R^{\mathcal{A}} \subseteq R^{\mathcal{B}}$, then $[\![Q]\!]^{\mathcal{A}} \subseteq [\![Q]\!]^{\mathcal{B}}$.
2. For all instances $\mathcal{A}, \mathcal{B}$, if $R^{\mathcal{A}} \subseteq R^{\mathcal{B}}$, then $[\![Q]\!]^{\mathcal{A}} \supseteq [\![Q]\!]^{\mathcal{B}}$.

PROPOSITION 5.4. *Let Q be a nonmonotonic relational algebra query that is built using a relation R and constant relations. Then R occurs at least twice in Q.*

Proof. Assume a relational algebra query tree exists that expresses Q and in which R only occurs as a single leaf. Then the path from that leaf towards the root operation consists of unary operations and operations $Q_1 \ \theta \ Q_2$ where Q_1 contains R and Q_2 has only constant relations as leaves: Q_2 is constant. So $Q_1 \ \theta \ Q_2$ can be thought of as a unary operation. But all unary operations θ are monotonic, i.e., if $X \subseteq Y$, then $\theta(X) \supseteq \theta(Y)$ for the family of operations $(C - X)_{C \text{ const.}, sch(C)=sch(X)}$ and $\theta(X) \subseteq \theta(Y)$ for all other operations. It follows that Q, a sequence of such operations, is also monotonic. $\square$

6. EXPRESSIVE POWER OF WSA WITHOUT DEFINITIONS

As the main technical result of the paper, we now show that WSA without definitions (but using repair-key as in our language definition), captures all of SO. It follows that definitions, despite our nondeterministic operations, do not add power to the language. This is surprising given Theorem 5.2.

The difficulty lies with the nondeterministic repair-key operation. If this operation occurs in a definition which is used more than once in the query, we cannot simply eliminate it by substitution, since multiple copies of a nondeterministic operation may produce different results. A definition evaluates to only a single result in each world – just like a (to be unambiguous, materialized) view.

We start by formalizing indicator relations, which will be an essential tool for rewriting queries to make sure that no view definition (or any other relation other than the domain relation) is used at more than one place in the query.

6.1 Indicator Relations

Let U be a *nonempty* relation (the *universe*) and let $R \subseteq U$. Then the indicator function $1_R : U \to \{0,1\}$ is defined as

$$1_R : x \mapsto \begin{cases} 1 & \dots \quad x \in R \\ 0 & \dots \quad x \notin R \end{cases}$$

The corresponding indicator relation is just the relation

$$\{\langle x, 1_R(x)\rangle \mid x \in U\}$$

which, obviously, has functional dependency $U \to \{0,1\}$. Subsequently, we will always use indicator relations rather than indicator functions and will denote them by 1_R as well. By our assumption that $U \neq \emptyset$, indicator relations are always nonempty.

Given relations R and U with $R \subseteq U \neq \emptyset$, the indicator relation 1_R w.r.t. universe U can be constructed in relational algebra as

$$\text{ind}(R, U) := (R \times \{\langle 1\rangle\}) \cup ((U - R) \times \{\langle 0\rangle\}).$$

The expression repair-key$_{sch(U)}(U \times \{0,1\})$ is equivalent to

$$\text{let } R := \text{subset}(U) \text{ in ind}(R, U)$$

and yields an indicator relation in each possible world.

Indicator relations have the nice property that their complement can be computed using a conjunctive query (with an inequality),

$$1_{U-R} = (U \times \{0,1\}) - 1_R := \pi_{1,2}(\sigma_{1=3 \wedge 2\neq 4}(U \times \{0,1\} \times 1_R)).$$

Let $\overline{R}$ denote the complement of relation R and let $U_i = R_i \cup \overline{R_i}$, called the *universe* of R_i. Note that

$$\overline{R_1 \times \cdots \times R_k} = \bigcup_{i=1}^{k} U_1 \times \cdots \times U_{i-1} \times \overline{R_i} \times U_{i+1} \times \cdots \times U_k.$$

The complement of a product $\vec{1} := 1_{R_1} \times \cdots \times 1_{R_k}$ can be obtained as

$$\begin{aligned} \text{compl}_{U_1,\dots,U_k}(\vec{1}) &= (U_1 \times \{0,1\} \times \cdots \times U_k \times \{0,1\}) - \vec{1} \\ &= \pi_{A_1,B_1,\dots,A_k,B_k}(\sigma_{\bigvee_i (A_i = A_i' \wedge B_i \neq B_i')}(\\ &\quad \rho_{A_1' B_1' \dots A_k' B_k'}(\vec{1}) \ \times \rho_{A_1 B_1 \dots A_k B_k}(\\ &\qquad U_1 \times \{0,1\} \times \cdots \times U_k \times \{0,1\}))). \end{aligned}$$

if, for each $1 \leq i \leq k$, U_i is the universe of R_i. Moreover,

LEMMA 6.1. *The k-times product of 1_R, denoted by*

$$(1_R)^k_U := \overbrace{1_R \times \cdots \times 1_R}^{k \text{ times}},$$

can be expressed as a relational algebra expression in which 1_R only occurs once.

Proof. Let U be the universe of R.

$$\begin{aligned} (1_R)^k_U &= \rho_{A_1 B_1 \dots A_k B_k}((U \times \{0,1\})^k) - \text{compl}_{U^k}(1^k_R) \\ &= \rho_{A_1 B_1 \dots A_k B_k}((U \times \{0,1\})^k) \\ &\quad - \pi_{A_1,B_1,\dots,A_k,B_k}(\sigma_{\bigvee_{1\leq i\leq k}(A_i = A' \wedge B_i \neq B')}(\\ &\qquad \rho_{A_1 B_1 \dots A_k B_k}((U \times \{0,1\})^k) \times \rho_{A' B'}(1_R))). \end{aligned}$$

$\square$

As a convention, let $S^0 = \{\langle\rangle\}$ for nonempty relations S. In particular, $(1_R)^0_U = \{\langle\rangle\}$.

6.2 The Quantifier-Free Case

By quantifier-free formulae we will denote formulae of predicate logic that have neither first- nor second-order quantifiers. An inequality (atom) is an atomic formula of the form $t_1 \, \theta \, t_2$ where t_1 and t_2 are variables or constants and θ is $=$, $\neq$, $<$, or $\leq$.

LEMMA 6.2. *Let ϕ be a negation- and quantifier-free formula with relations $\vec{R}$. Then ϕ can be translated in linear time into a formula $\exists\vec{x} \ \alpha \wedge \beta$, where α is a Boolean combination of inequalities and β is a conjunction of relational literals, which is equivalent to ϕ on structures in which each relation of $\vec{R}$ is nonempty.*

Proof. Let $R_1, \dots, R_s$ the set of distinct predicates (relation names) occurring in ϕ. Apply the following translation of ϕ inductively bottom-up.

Induction start: The translation is the identity on inequality atoms. Rewrite atomic formulae $R_j(\vec{t})$ into $\exists \vec{v}_{j1}\ \vec{v}_{j1} = \vec{t} \wedge R_j(\vec{v}_{j1})$. The free variables of these formulae are the variables occurring in $\vec{t}$. The variables $\vec{v}_{j1}$ are new and do not overlap with those of $\vec{t}$.

Induction step: A subformula $\psi_1 \vee \psi_2$ (resp., $\psi_1 \wedge \psi_2$) with

$$\psi_i = \exists \vec{v}\vec{w}\ \ \alpha_i \wedge \bigwedge_{j=1}^{s} \bigwedge_{k=1}^{n_{ij}} R_j(\vec{v}_{jk}).$$

is turned into

$$\exists \vec{v}\vec{w}\ \ \alpha \wedge \bigwedge_{j=1}^{s} \bigwedge_{k=1}^{m_j} R_j(\vec{v}_{jk})$$

where $m_j = \max(n_{1j}, n_{2j})$ and $\alpha = \alpha_1 \vee \alpha_2$ (resp., $m_j = n_{1j} + n_{2j}$, $\alpha = \alpha_1 \wedge \alpha_2'$, and α_2' is obtained from α_2 by replacing each occurrence of variable $v_{j,k,l}$ everywhere in α_2' by $v_{j,k+n_{1j},l}$).

For the equivalence of the rewritten formula to ϕ, it is only necessary to point out that since, by assumption of the lemma, all the relations R_j are nonempty, ψ_i is equivalent to

$$\exists \vec{v}\vec{w}\ \ \alpha_i \wedge \bigwedge_{j=1}^{s} \bigwedge_{k=1}^{m_j} R_j(\vec{v}_{jk}).$$

It is not hard to verify that the translation can indeed be implemented to run in linear time and that the rewritten formula is of the form claimed in the lemma. □

Theorem 6.3. *For any quantifier-free formula there is an equivalent positive relational algebra expression over universe relations and indicator relations in which each indicator relation only occurs once.*

Proof Sketch. Assume $R_1, \dots, R_s$ are all the predicates (or equivalently, second-order variables) occurring in the formula, and $\vec{x}$ are all the free first-order variables of the formula.

First push negations in the formula down to the atomic formulae using De Morgan's laws and the elimination of double negation. Then replace each relational literal $\neg R_j(\vec{t})$ by the equivalent atom $1_{R_j}(\vec{t}, 0)$, and replace each remaining relational atom $R_j(\vec{t})$ by the equivalent atom $1_{R_j}(\vec{t}, 1)$. Moreover, we replace each inequality literal $\neg(t\,\theta\,t')$ by $t\,\theta'\,t'$, where θ' is the complement of θ (e.g., $\neq$ for $=$).

Since $1_{R_j} \neq \emptyset$ and our formula is now negation-free, we can use Lemma 6.2 to rewrite the formula into a formula of syntax

$$\exists \vec{v}\vec{w}\ \alpha \wedge \bigwedge_{j=1}^{s} \bigwedge_{k=1}^{m_j} 1_{R_j}(\vec{v}_{jk})$$

where α does not contain relational atoms.

The relational algebra expression is $\pi_{\vec{x}}(\sigma_\alpha(B_1 \times \cdots \times B_s))$ with $B_j := \rho_{\vec{v}_{j1}\dots\vec{v}_{jm_j}}\big((1_{R_j})^{m_j}_{U_j}\big)$. Each B_j computes an m_j-times product of 1_{R_j} using the technique of Lemma 6.1 which just uses one occurrence of 1_{R_j}. All the relations 1_{R_j} only occur once. This proves the theorem. □

Example 6.4. Consider an alternative encoding of 3-colorability in WSA which is based on guessing a subset of relation $U = V \times \rho_C(\{r, g, b\})$. Then 3-colorability is the problem of deciding the SO sentence $\exists C(\subseteq U)\ \neg\exists v, w, c, c'\ \phi_1 \vee \phi_2 \vee \phi_3$ with $\phi_1 = E(v, w) \wedge C(v, c) \wedge C(w, c)$, $\phi_2 = C(v, c) \wedge C(v, c') \wedge c \neq c'$, and $\phi_3 = \neg C(v, r) \wedge \neg C(v, g) \wedge \neg C(v, b)$, i.e., ϕ_1 asserts that two neighboring nodes have the same color, ϕ_2 that a node has simultaneously two colors, and ϕ_3 that a node has not been assigned any color at all. If neither is the case, we have a 3-coloring of the graph. Using Theorem 6.3, $\phi_1 \vee \phi_2 \vee \phi_3$ becomes

$$\pi = \exists t_1 \dots t_4\ (\psi_1 \vee \psi_2 \vee \psi_3) \wedge 1_C(u_1, c_1, t_1) \wedge 1_C(u_2, c_2, t_2) \wedge 1_C(u_3, c_3, t_3) \wedge 1_E(v, w, t_4)$$

where

$$\begin{aligned}
\psi_1 &= u_1 = v \wedge u_2 = w \wedge c_1 = c_2 \wedge t_1 = t_2 = t_4 = 1 \\
\psi_2 &= u_1 = u_2 \wedge c_1 \neq c_2 \wedge t_1 = t_2 = 1 \\
\psi_3 &= u_1 = u_2 = u_3 \wedge c_1 = r \wedge c_2 = g \wedge c_3 = b \\
&\wedge\ \ t_1 = t_2 = t_3 = 0;
\end{aligned}$$

This is a slight simplification of the translation result we would obtain following the proof of Lemma 6.2: We do not define copies of the free variables just to quantify them away again.

Following Theorem 6.3, formula π can be turned into relational algebra as

$$Q_\pi := \pi_{u_1c_1u_2c_2u_3c_3vw}(\sigma_{\psi_1 \vee \psi_2 \vee \psi_3}(\rho_{u_1c_1t_1u_2c_2t_2u_3c_3t_3}((1_C)^3_{V \times \{r,g,b\}}) \times \rho_{vwt_4}(E)))$$

where $(1_C)^3_{V \times \{r,g,b\}}$ denotes the relational algebra expression for $1_C \times 1_C \times 1_C$ from Lemma 6.1.

The complete SO sentence can be stated as

$$\exists 1_C\ (1_C : V \times \{r, g, b\} \to \{0, 1\}) \wedge \neg\exists u_1c_1u_2c_2u_3c_3vw\ \pi.$$

If 1_C in Q_π is replaced by repair-key$_{V,C}(V \times \rho_C(\{r, g, b\}) \times \rho_T(\{0, 1\}))$, this sentence can be turned into WSA without definitions as possible$(\{\langle\rangle\} - \pi_\emptyset(Q_\pi))$. □

6.3 Quantification and Alternation

Conceptually, in SO, there is no difference in the treatment of second-order variables and relations coming from the input structure; an existential second-order quantifier extends the structure over which the formula is evaluated. In our algebra, however, we have to construct the possible alternative relations for a second-order variable R at the beginning of the bottom-up evaluation of the algebra expression using repair-key and have to later test the existential quantifier $\exists R$ using the possible operation grouping the possible worlds that agree on R. For that we have to keep R around during the evaluation of the algebra expression. Selections also must not actually remove tuples because this would mean that the information about which world the tuple is missing from would be lost. For example, the algebra expression corresponding to a Boolean formula must not return false, but in some form must compute the pair $\langle R, \text{false}\rangle$.

Let ϕ be an SO formula with free second-order variables $R_1, \dots, R_k$ and l free first-order variables $x_1, \dots, x_l$. Conceptually, our proofs will produce a WSA expression for ϕ that computes, in each possible world identified by choices of relations $R_1, \dots, R_k$ for the free second-order variables, the relation

$$R_1 \times \cdots \times R_k \times \Theta$$

where Θ is a representation of a mapping

$$\vec{a} \mapsto \text{truth value of } (R_1, \dots, R_k) \vDash \phi[\vec{x} \text{ replaced by } \vec{a}].$$

Truth and falsity cannot be just represented by 1 and 0, respectively, because an existential first-order quantifier will effect a projection on Θ whose result may contain both truth values 1 and 0 for a variable assignment $\vec{a}$. Thus, projection may map environments for which ϕ is true together with environments for which ϕ is false. In that case we would like to remove the tuples for which the truth value encoding is 0. Unfortunately, the function

$$F : \begin{cases} \{0\} \mapsto \{0\} \\ \{1\} \mapsto \{1\} \\ \{0,1\} \mapsto \{1\} \end{cases}$$

is nonmonotonic, and by Proposition 5.4 cannot be expressed in relational algebra if the input relation is to occur in the query only once. Fortunately, we do not need such a function F.

Definition 6.5. *A PBIT (protected bit) is either $\{\bot\}$ (denoting 0) or $\{\bot, 1\}$ (denoting 1).*

Given a Boolean query Q (i.e., Q returns either $\{\langle\rangle\}$ or $\emptyset$),

$$PBIT(Q) := (Q \times \{1\}) \cup \{\bot\}.$$

The negation of PBIT B is obtained by $\{\bot, 1\} - (B \cap \{1\})$. The set union on PBITs effects a logical OR, thus a relation $\subseteq R \times PBIT$ for which $\langle \vec{a}, 1\rangle \in R$ implies $\langle \vec{a}, \bot\rangle \in R$ guarantees that projecting away a column other than the rightmost corresponds to existential quantification.

Definition 6.6 (Protected truth-table semantics). *Under the protected truth-table semantics, a second-order formula ϕ with free second-order variables $R_1, \dots, R_k$ and free first-order variables $x_1, \dots, x_l$, computes on input relational database $\mathcal{A}$ the relation*

$$[\![\phi]\!]^{\mathcal{A}}_{tt}(R_1, \dots, R_k) := 1_{R_1} \times \cdots \times 1_{R_k} \times \Theta$$

such that

$$\Theta = \rho_{\vec{D},T}((D^l \times \{\bot\}) \cup \{\langle \vec{a}, 1\rangle \mid \vec{a} \in D^l, \\ (\mathcal{A}, R_1, \dots, R_k) \vDash \phi[\vec{x} \text{ replaced by } \vec{a}] \text{ is true}\})$$

and D is a domain relation containing the possible values for the first-order variables.

A relation Θ can therefore be thought of as a mapping $D^l \to PBIT$. The complement of such a relation Θ is

$$\text{compl}_{D^l}(\Theta) := D^l \times \{\bot, 1\} - \sigma_{T=1}(\Theta).$$

Example 6.7. Let $R = \{\langle 0, 1\rangle, \langle 1, 0\rangle\}$ with schema $R(AB)$ and let $\phi = R(x, 1)$. Then

$$[\![\phi]\!]^{()}_{tt}(R) = \begin{array}{c|ccc} 1_R & A & B & (\in R) \\ \hline & 0 & 0 & 0 \\ & 0 & 1 & 1 \\ & 1 & 0 & 1 \\ & 1 & 1 & 0 \end{array} \times \begin{array}{c|cc} \Theta & x & T \\ \hline & 0 & 1 \\ & 0 & \bot \\ & 1 & \bot \end{array}$$

because $PBIT(R \vDash \phi[0]) = \{1, \bot\}$ and $PBIT(R \vDash \phi[1]) = \{\bot\}$. □

We will define WSA expressions that correspond to SO formulae under the protected truth-table semantics.

Next we obtain an auxiliary construction for complementing a Θ relation while passing on the values of the second-order variables. This will be the essential tool for alternation.

Lemma 6.8. *Let $P = 1_{R_1} \times \cdots \times 1_{R_k} \times \Theta$ where $\Theta \subseteq D_1 \times \cdots \times D_l \times PBIT$. There is a WSA expression without definitions for*

$$\text{compl}_{U_1,\dots,U_k;\vec{D},T}(P) := 1_{R_1} \times \cdots \times 1_{R_k} \times \text{compl}(\Theta)$$

in which P only occurs once.

Proof. Let $sch(U_i) = A_i$ and $sch(1_{R_i}) = A_i B_i$. We write $\vec{1}$ for $1_{R_1} \times \cdots \times 1_{R_k}$ and $\vec{U}^+$ for $U_1 \times \cdots \times U_k \times \rho_{B_1 \dots B_k}(\{0,1\}^k)$. An encoding of $\text{compl}_{U_1,\dots,U_k}(\vec{1})$ in relational algebra was given in Section 6.1.
$\text{compl}_{U_1,\dots,U_k;\vec{D},T}(\vec{1} \times \Theta) =$

$$\begin{aligned} & \vec{1} \times (D^l \times \{\bot, 1\} - \sigma_{T=1}(\Theta)) \\ = \;& (\vec{U}^+ \times D^l \times \rho_T(\{\bot, 1\})) \\ & \quad -\text{compl}_{U_1,\dots,U_k}(\vec{1}) \times D^l \times \rho_T(\{\bot, 1\}) \\ & \quad -\vec{U}^+ \times \sigma_{T=1}(\Theta) \\ = \;& (\vec{U}^+ \times D^l \times \rho_T(\{\bot, 1\})) \\ & -\pi_{A_1,B_1,\dots,A_k,B_k,T}(\sigma_{\bigvee_i (A_i = A'_i \wedge B_i \neq B'_i) \vee T' = T = 1}(\\ & \quad \vec{U}^+ \times \rho_{A'_1 B'_1 \dots A'_k B'_k T'}(\underbrace{\vec{1} \times \Theta}_{P}) \times \rho_T(\{\bot, 1\}))). \end{aligned}$$

The final WSA expression is in the desired form. □

Now we are ready to prove the main result of this section.

Theorem 6.9. *Given a second-order formula ϕ, a WSA expression without definitions which is equivalent to ϕ under the protected truth-table semantics can be computed in linear time in the size of ϕ.*

Proof Sketch. The proof is by bottom-up induction in the formula. We will construct for each SO formula ϕ with free second-order variables $R_1, \dots, R_k$ on input structure $\mathcal{A}$ (that is, the input relations from $\mathcal{A}$ are *separate* from $R_1, \dots, R_k$) a WSA expression P such that

$$[\![P]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} = \{[\![\phi]\!]^{\mathcal{A}}_{tt}(R_1, \dots, R_k) \mid R_1 \subseteq U_1, \dots, R_k \subseteq U_k\}.$$

Induction start: Assume that ϕ is quantifier-free with free second-order variables $R_1, \dots, R_k$ ($k \geq k_0$) and free first-order variables $\vec{x}$. Consider the quantifier-free formula

$$\psi(\vec{y}, \vec{x}, t) := \Big(\bigwedge_{1 \leq j \leq k} 1_{R_j}(\vec{y}_j) \Big) \wedge \big((\phi \wedge t = 1) \vee t = \bot \big),$$

where the variables $\vec{y}$ and t are new and do not occur in ϕ. It is easy to verify that ψ defines the relation $[\![\phi]\!]^{\mathcal{A}}_{tt}(R_1, \dots, R_k)$ in the sense that

$$(R_1, \dots, R_k) \vDash \psi[\vec{t}] \;\Leftrightarrow\; \vec{t} \in [\![\phi]\!]^{\mathcal{A}}_{tt}(R_1, \dots, R_k).$$

Specifically, the projection down to columns $\vec{y}_j$ represents the free second-order variable R_j, the projection down to columns $\vec{x}$ specifies all the possible assignments to the first-order variables, and t is a PBIT for the truth value of ϕ for a given assignment to the first- and second-order variables.

The corresponding WSA expression without definitions P is obtained as follows. We first translate ψ into a relational algebra expression using Theorem 6.3. In this expression, each relation 1_{R_j} occurs at most once. Apart from that, it uses only universe relations U_j (which can be thought of as $ar(R_j)$-times products of D). In this expression, we replace each indicator relation 1_{R_j} as follows. For input relations R_j ($j \leq k_0$), we replace 1_{R_j} by the algebra expression $\text{ind}(R_j, U_j)$. For second-order variables R_j ($j > k_0$), we replace 1_{R_j} by $\text{repair-key}_{sch(U_j)}(U_j \times \{0,1\})$. The claim of the induction hypothesis follows immediately.

Induction step (ϕ has quantifiers): We assume that universal quantifiers $\forall \cdot$ have been replaced by $\neg\exists\cdot\neg$.

- First-order existential quantification: $\phi = \exists x_{l+1}\ \psi$, where the free first-order variables of ψ are $x_1, \dots, x_{l+1}$. By the induction hypothesis, there is a WSA expression P which computes $[\![\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k)$. The corresponding WSA expression is $\pi_{sch(P)-x_{l+1}}(P)$. It is easy to verify that the projection has exactly the effect of existential first-order quantification,
$$[\![\exists x_{l+1}\ \psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k) \ = \ \pi_{sch(P)-x_{l+1}}([\![\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k)).$$
The semantics of WSA ensures that this projection is performed in parallel in all possible worlds (i.e., for any interpretation of the free second-order variables we may choose), without interference (see Lemma 3.5 and Lemma 3.6).

- Second-order existential quantification: $\phi = \exists R_{k+1}\ \psi$. By the induction hypothesis, there is a WSA expression P which computes $[\![\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_{k+1})$. We may assume w.l.o.g. that the relations $R_1,\dots,R_{k+1}$ have disjoint schemas. The WSA expression for ϕ is
$$Q = \pi_{sch(P)-sch(R_{k+1})}(\text{possible}_{sch(R_1),\dots,sch(R_k)}(P)).$$
The correctness is established as follows.
$$\begin{aligned}
[\![Q]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} &:= \\
&\Big\{\pi_{sch(P)-sch(R_{k+1})}\Big(\bigcup\{S \in [\![P]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} \\
&\qquad \mid \pi_{sch(R_1),\dots,sch(R_k)}(S) = 1_{R_1} \times \cdots \times 1_{R_k}\}\Big) \\
&\qquad\qquad \mid \vec{R} \subseteq \vec{U}\Big\} \\
&= \Big\{\pi_{sch(P)-sch(R_{k+1})}\Big(\\
&\qquad \bigcup_{R_{k+1}\subseteq U_{k+1}} [\![\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_{k+1})\Big) \mid \vec{R} \subseteq \vec{U}\Big\} \\
&= \big\{[\![\exists R_{k+1}\ \psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k) \mid \vec{R} \subseteq \vec{U}\big\}
\end{aligned}$$
where $\vec{R} \subseteq \vec{U}$ is a shortcut for $R_1 \subseteq U_1, \dots, R_k \subseteq U_k$.

- Negation: $\phi = \neg\psi$. The WSA expression for ϕ is $\text{compl}_{U_1\dots U_k;\vec{D},T}(P)$. By Lemma 6.8,
$$[\![\neg\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k) \ = \ \text{compl}_{U_1\dots U_k;\vec{D},T}([\![\psi]\!]^{\mathcal{A}}_{tt}(R_1,\dots,R_k)).$$
The correctness follows from the fact that the operations of $\text{compl}_{U_1\dots U_k;\vec{D},T}(\cdot)$ are relational algebra, which is evaluated in WSA in each world in parallel.

For an SO formula ϕ without free second-order-variables, the WSA expression P computes a singleton relation
$$[\![P]\!]^{\mathcal{A}}_{\{\mathcal{A}\}} = \{[\![\phi]\!]^{\mathcal{A}}_{tt}()\},$$
that is, there is a single possible world as result. $\square$

Remark 6.10. *From protected truth-table to standard semantics of SO:* Let ϕ be an SO formula without free second-order variables (but possibly with input relations) and let P be the corresponding WSA expression given by Theorem 6.9. The relation defined by the free first-order variables of ϕ, $\{\vec{a} \in D^l \mid \mathcal{A} \vDash \phi[\vec{a}]\}$, is obtained as $\pi_{sch(P)-\{T\}}(\sigma_{T=1}(P))$.

Note that the proof construction of Theorem 6.9 is stronger: If ϕ has free second-order variables, then the corresponding WSA expression computes, for each possible interpretation of the free second-order variables, a possible world with the relation defined by the first-order variables and the values taken by the second-order variables all encoded in a single unambiguous relation according to $[\![\cdot]\!]_{tt}$. $\square$

Example 6.11. We continue Example 4.2. Let
$$\phi = \big(L(c,p,0) \wedge (P_1(p) \vee P_2(p))\big) \vee \big(L(c,p,1) \wedge \neg(P_1(p) \vee P_2(p))\big).$$
Then Σ_2-QBF can be expressed by the SO sentence
$$\exists P_1(\subseteq V_1)\ \neg\exists P_2(\subseteq V_2)\ \neg\exists c(\in C)\ \neg\exists p\ \phi.$$
We can turn $\big(\phi \vee t = \bot\big) \wedge P_1(p_{12}) \wedge P_2(p_{22})$ into WSA over indicator relations as
$$Q = \sigma_\psi\big(\rho_{cpst_L}(1_L) \times \rho_{p_{11}t_{11}p_{12}t_{12}}((1_{P_1})^2_{V_1}) \times \rho_{p_{21}t_{21}p_{22}t_{22}}((1_{P_2})^2_{V_2}) \times \rho_t(\{\langle\bot\rangle\} \cup \{\langle 1\rangle\})\big)$$
where $\psi = \big(t = \bot \vee (t_L = 1 \wedge p = p_{11} = p_{21} \wedge ((s = 0 \wedge (t_{11} = 1 \vee t_{21} = 1)) \vee (s = 1 \wedge t_{11} \neq 1 \wedge t_{21} \neq 1)))\big)$. Note that we have simplified the expression of the proof somewhat by inlining the auxiliary variables $\vec{v}$ and $\vec{w}$.

The complete WSA expression for the SO sentence is
$$\overbrace{\pi_\emptyset \circ \sigma_{t=1}}^{PBIT \text{ to bool}} \circ \overbrace{\pi_t \circ \text{possible}_\emptyset}^{\exists P_1} \circ \overbrace{\text{compl}_{V_1;T}}^{\neg} \circ \underbrace{\pi_{p_{12}t_{12}t} \circ \text{possible}_{p_{12}t_{12}}}_{\exists P_2} \circ \underbrace{\text{compl}_{V_1,V_2;T}}_{\neg} \circ \underbrace{\pi_{p_{12}t_{12}p_{22}t_{22}t}}_{\exists c} \circ \underbrace{\text{compl}_{V_1,V_2;C,T}}_{\neg} \circ \underbrace{\pi_{p_{12}t_{12}p_{22}t_{22}ct}}_{\exists p}(\underbrace{\ Q\ }_{\phi}).$$
We replace 1_L by $\text{ind}(L,\cdot)$ and 1_{P_i} by $\text{repair-key}_p(\rho_{pt}(V_i \times \{0,1\}))$. $\square$

The composition of Theorems 4.3 and 6.9 yields a translation[3] from WSA with definitions to WSA without. Thus, definitions add no power to WSA.

Corollary 6.12. *WSA without definitions captures WSA.*

[3]This translation is inefficient, however: It may produce WSA expressions that are exponentially larger than the input, and may introduce additional difference operations.

7. COMPLEXITY OF WSA

The data complexity of a query language refers to the problem of evaluating queries on databases assuming the queries fixed and only the database part of the input, while combined complexity assumes that both the query and the database are part of the input [19].

By [18], a generalization of Fagin's Theorem [8] (see also [15]), the data complexity of SO logic *corresponds exactly* to the polynomial hierarchy (PHIER) in the following way: The data complexity of each SO sentence is in one of the classes of the polynomial hierarchy, and for each such class there is an SO sentence whose data complexity is complete for it. (The verbose definition is necessary because no individual problem is complete for PHIER unless the hierarchy collapses.) SO logic is also PSPACE-complete with respect to combined complexity.

By the interreducibility between SO and WSA, which is efficient for the reduction from SO to WSA without definitions (Theorem 6.9, which implies PSPACE-hardness of WSA with or without definitions) and the reduction from WSA without definitions to SO (Theorem 4.3, which implies PSPACE-membership of WSA without definitions), the following complexity results follow for WSA.

Corollary 7.1. *1. WSA with or without definitions corresponds exactly to PHIER with respect to data complexity,*

2. WSA with definitions is PSPACE-hard with respect to combined complexity, and

3. WSA without definitions is PSPACE-complete with respect to combined complexity.

We cannot directly conclude an upper bound on the combined complexity of WSA with definitions from the reduction of Theorem 4.3 because it takes exponential time: In the case that WSA definitions are used, several copies of formulae ψ_V may be used in the SO formula constructed in the proof. By inductively expanding the formula, eliminating definitions, we cause an exponential blow-up of formula size. However, we can think of the proof construction as a linear-time mapping from WSA with definitions to second-order logic with definitions. The standard PSPACE algorithm for second-order logic extends directly to second-order logic with definitions: Of the formula, we only have to maintain a current path in its parse tree, which is clearly of polynomial size. It follows that

Proposition 7.2. *WSA with definitions is PSPACE-complete with respect to combined complexity.*

8. RELATED WORK

In an early piece of related work, Libkin and Wong [16] define a query algebra for handling both nested data types and uncertainty. Their notion of uncertainty called *or-sets* (as a generalization of the or-sets of [11]) is treated as a special collection type that can syntactically be thought of as a set of data and is only interpreted as uncertainty on an additional "conceptual level". The result is a very elegant and clean algebra that nicely combines complex objects with uncertainty. While their language is stronger and can manage nested data, there is nevertheless a close connection to WSA, which can be thought of as a flat relational version of their language. Indeed, the or-set language contains an operator α that is essentially equivalent to the repair-key operator of WSA.

TriQL, the query language of the Trio project [20], subsumes the power of relational algebra and supports an operation "groupalts" which expresses the repair-key operation of WSA applied to a certain relation. There are many more operations in TriQL, but it is hard to tell whether possible$_{\vec{A}}$ is expressible in TriQL since no formal semantics of the language is available. Moreover, explicitly processing and modifying the Trio representation system is central to TriQL's design philosophy. As a consequence, TriQL contains a number of representation-dependent (non-generic [2]) operations which may return semantically different results for different semantically equivalent representations of a probabilistic database. This makes TriQL hard to study and compare with WSA. However, it seems that WSA is a good candidate for a generic core to TriQL, and the results of the present paper provide additional evidence that TriQL is highly expressive.

The probabilistic databases definable using repair-key from certain relations are also a large fragment of the block independent-disjoint (BID) tables of Ré and Suciu: Using repair-key and selection, all the BID tables can be defined. In their paper [17], they study related representability problems for BID tables. BID tables are known to be more powerful than tuple-independent tables, which correspond to uncertain tables definable using the subset operation. This is in line with observations made in Section 5 of the present paper.

In [7], Dalvi and Suciu characterize a class of conjunctive queries that are #P-hard with self-joins and in polynomial time without, namely the hierarchical queries with an inversion that does not have an eraser (for a definition of that class, see [7]). The present paper shows that self-joins in relational algebra can be essentially eliminated at the cost of introducing difference operations. This directly yields a #P-hardness result for a very restrictive class of queries with difference on probabilistic databases.

The algebra defined in our earlier work [4] is exactly the one described in the present paper, modulo the following details. Most importantly, while repair-key is introduced there as part of the algebra, most of the paper focuses on the fragment that is obtained by replacing repair-key by choice-of. Moreover, the syntax of possible$_{\vec{A}}$ allows for the grouping of worlds by a query Q that can be given as a parameter; the syntax is possible$_Q(Q')$. An operation possible$_{\vec{A}}$ in the syntax of the present paper corresponds to an operation possible$_{\pi_{\vec{A}}}$ in the syntax of [4]. The results of this paper imply that allowing general queries Q for grouping adds no power, so we are indeed studying the same language. The paper [4] also gives an SQL-like syntax for WSA, in which the intuition of possible$_{\vec{A}}$ is made explicit by the syntax "select possible ... group worlds by ...".

In recent work [3, 14, 12], efficient techniques for processing a large part of WSA have been developed. The only operations that currently defy good solutions are possible$_{\vec{A}}$ (i.e., with world grouping) and, to a lesser extent, relational difference. Indeed, the repair-key operator on the standard representations described in Example 2.1 can be implemented efficiently, even though semantically it generally causes an exponential blowup in the size of the set of possible worlds. Thus, it is natural to ask for the expressive power of WSA with possible$_{\vec{A}}$ replaced by possible. The construction of the proof of Theorem 4.1 can map any SO formula of the form $\exists R\ \phi$ or $\forall R\ \phi$, where ϕ is FO, to WSA. It is not hard to see that despite the restriction to a *single* second-order quantifier, this fragment of WSA (with definitions) can express all of NP $\cup$ co-NP. For an upper bound, it seems that all such restricted WSA queries have data complexity in Δ_2^P (i.e., P^{NP}).

9. CONCLUSIONS

The main contribution of this paper is to give the apparently first compositional algebra that exactly captures second-order logic over finite structures, a logic of wide interest.

Second-order logic is a natural yardstick for the expressiveness of query languages for uncertain databases. It is an elegant and well-studied formalism that naturally captures what-if queries. It can be argued that second-order logic takes the same role in uncertain databases that first-order logic and relational algebra take in classical relational databases. In that sense, the expressiveness result of this paper, $WSA = SO$, is an uncertain databases analog of Codd's Theorem.

Finding the right query algebra for uncertain databases is important because efficient query processing techniques are easier to obtain for algebraic languages without variables or quantifiers, and algebraic operators are natural building blocks for database query plans. Of course, the expressiveness result of this paper also implies that WSA has high complexity and thus this paper can only be an initial call for the search for more efficiently processible fragments of WSA that retain some of its flavor of simplicity and cleanliness.

Acknowledgments

The author thanks Michaela Goetz and the anonymous reviewers of ICDT 2009 for their useful suggestions that helped to improve this paper. This work was supported by NSF grant IIS-0812272 and a KDD grant.

10. REFERENCES

[1] S. Abiteboul and C. Beeri. "The Power of Languages for the Manipulation of Complex Values". *VLDB J.*, **4**(4):727–794, 1995.

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

[3] L. Antova, T. Jansen, C. Koch, and D. Olteanu. "Fast and Simple Relational Processing of Uncertain Data". In *Proc. ICDE*, 2008.

[4] L. Antova, C. Koch, and D. Olteanu. "From Complete to Incomplete Information and Back". In *Proc. SIGMOD*, 2007.

[5] O. Benjelloun, A. D. Sarma, A. Halevy, and J. Widom. "ULDBs: Databases with Uncertainty and Lineage". In *Proc. VLDB*, 2006.

[6] N. Dalvi and D. Suciu. "Efficient query evaluation on probabilistic databases". *VLDB Journal*, **16**(4):523–544, 2007.

[7] N. Dalvi and D. Suciu. "The Dichotomy of Conjunctive Queries on Probabilistic Structures". In *Proc. PODS*, pages 293–302, 2007.

[8] R. Fagin. "Generalized First-Order Spectra and Polynomial-Time Recognizable Sets". In *Complexity of Computation, R. Karp, ed., SIAM-AMS Proceedings 7*, pages 27–41. 1974.

[9] J. Huang, D. Olteanu, and C. Koch. "SPROUT: Lazy vs. Eager Query Plans for Tuple-Independent Probabilistic Databases". In *Proc. ICDE*, 2009. To appear.

[10] T. Imielinski and W. Lipski. "Incomplete information in relational databases". *Journal of the ACM*, **31**(4):761–791, 1984.

[11] T. Imielinski, S. Naqvi, and K. Vadaparty. "Incomplete objects — a data model for design and planning applications". In *Proc. SIGMOD*, pages 288–297, 1991.

[12] C. Koch. "Approximating Predicates and Expressive Queries on Probabilistic Databases". In *Proc. PODS*, 2008.

[13] C. Koch. "MayBMS: A system for managing large uncertain and probabilistic databases". In C. Aggarwal, editor, *Managing and Mining Uncertain Data*, chapter 6. Springer-Verlag, 2008. To appear.

[14] C. Koch and D. Olteanu. "Conditioning Probabilistic Databases". In *Proc. VLDB*, 2008.

[15] L. Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[16] L. Libkin and L. Wong. "Semantic Representations and Query Languages for Or-Sets". *J. Comput. Syst. Sci.*, **52**(1):125–142, 1996.

[17] C. Ré and D. Suciu. "Materialized Views for Probabilistic Databases (For Information Exchange and Query Optimization)". In *Proc. VLDB*, 2007.

[18] L. Stockmeyer. "The Polynomial Hierarchy". *Theor. Comput. Sci.*, **3**:1–22, 1977.

[19] M. Y. Vardi. "The Complexity of Relational Query Languages". In *Proc. STOC*, pages 137–146, 1982.

[20] J. Widom. "TRIO: A System for Managing Data, Uncertainty, and Lineage". In C. Agarwal, editor, *Managing and Mining Uncertain Data*, 2008. To appear.