On Enumerating Query Plans Using Analytic Tableau

Alexander Hudek, David Toman, and Grant Weddell

Cheriton School of Computer Science
University of Waterloo, Canada
{akhudek, david, gweddell}@uwaterloo.ca

Abstract. We consider how the method of analytic tableau coupled with interpolant extraction can be adapted to enumerate possible query plans for a given user query in the context of a first order theory that defines a relational database schema. In standard analytic tableau calculi, the sub-formula property of proofs limits the variety of interpolants and consequently of plans that can be generated for the given query. To overcome this limitation, we present a two-phase adaptation of a tableau calculus that ensures all plans logically equivalent to the query with respect to the schema, that correctly implement the user query, are indeed found. We also show how this separation allows us to avoid backtracking when reasoning about consequences of the schema.

1 Introduction

First order logic (FOL) lies at the heart of a relational database system (RDBMS) and has had a profound influence on the development of its interface in which users express queries over a *logical design*, a domain specific ontological appreciation of relevant data: witness relational algebra and the SQL query language. Indeed, relational technology is a multi-billion dollar industry, and constitutes one of the most successful influences of logic in computer science.

In an RDBMS, there is a fundamental and crucial distinction between a logical design and a *physical design*. The latter refines the former with mapping rules that relate logical artifacts to various material capabilities for accessing data. The contents of various data structures such as records, arrays, files and ordered indices and the contents of legacy data managed by a separate RDBMS are examples of such capabilities.

Typically, a user query over a logical design will have a large number of potential query execution plans over a physical design, and it is not uncommon for these plans to differ by orders of magnitude in their efficiency. It is therefore imperative that the query optimizer of an RDBMS is able to enumerate possible plans with reasonable efficiency, a requirement that has become more challenging with recent trends in information integration: view based query rewriting, ontology based data access, main memory databases, and so on. Such trends have made the relationships between a logical design and the material capabilities for

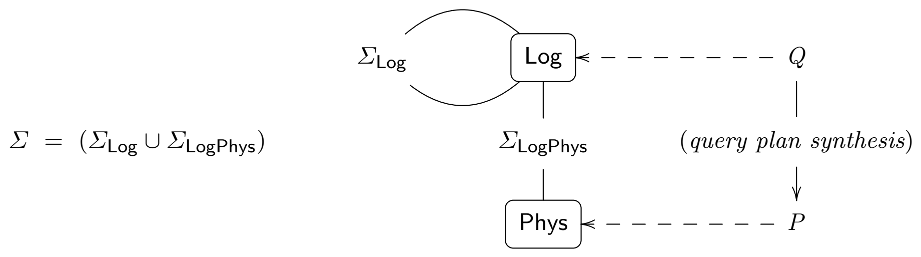
Σ_{Log}
Log
Q
$\Sigma = (\Sigma_{\mathsf{Log}} \cup \Sigma_{\mathsf{LogPhys}})$
$\Sigma_{\mathsf{LogPhys}}$
(*query plan synthesis*)
Phys
P

Fig. 1. Overview of plan synthesis for FOL queries.

accessing data manifest in a physical design much less straightforward and more indirect, and have therefore complicated the task of plan enumeration.

In this paper, we address the problem of plan enumeration in the context of a (function free) FOL theory Σ that serves as a very powerful language for capturing a physical design. Figure 1 summarizes the general problem in the FOL setting. Here, Q and P correspond, respectively, to a user query and to a query plan that implements the query. We assume both Q and P are range-restricted formulae in FOL expressed over distinct relational signatures Log and Phys. The former is a collection of predicate symbols that forms part of the logical design for a database, the set of "tables" visible to the user formulating Q. The latter is the above-mentioned collection of predicate symbols (often called *access paths*) abstracting the material capabilities for accessing data and ranging from scanning linked lists and navigating pointers in main memory to accessing external data sources [23]. In addition to Log, a logical design consists of sentences Σ_{Log} over Log that capture constraints such as keys and other forms of dependencies, view definitions, additional domain specific constraints, and so on. The physical design, denoted as Σ in the figure, augments Log with Phys and Σ_{Log} with additional sentences $\Sigma_{\mathsf{LogPhys}}$ over Log $\cup$ Phys. The additional sentences establish the mapping relationships between a logical design and the information sources given by Phys.

In this setting, we consider how the method of analytic tableau can be adapted to help enumerate possible query plans for a given user query when (1) the material capabilities for accessing data are given by a subset of the predicate symbols occurring in Σ, and when (2) both logical design and the mapping relationships to the access paths are captured by the formulae that comprise Σ.

Our contributions in this paper are fourfold:

1. We introduce a *conditional tableau* that abstracts reasoning common to *all* query plans for a given user query and physical design.
2. We show how the results of the conditional tableau phase *compactly summarize* the space of possible query plans.
3. We show how alternative query plan candidates can be generated and how their validity can be determined *without any additional tableau reasoning.*[1]

[1] Notwithstanding the possible need to incrementally extend the depth of the conditional tableau due to the computational nature of the problem.

$$\forall n, s_1, s_2, a_1, a_2.(\mathsf{Emp}(n, s_1, a_1) \land \mathsf{Emp}(n, s_2, a_2) \to ((s_1 = s_2) \land (a_1 = a_2)))$$
$$\forall n.((\exists s, a.\mathsf{Emp}(n, s, a)) \leftrightarrow (\mathsf{Mgr}(n) \lor \mathsf{Wkr}(n)))$$
$$\forall n.(\mathsf{Mgr}(n) \to \neg\mathsf{Wkr}(n))$$

(a) Logical design: Σ_{Log}

$$\forall n, s.(\mathsf{Sv}(n, s) \leftrightarrow (\exists a.\mathsf{Emp}(n, s, a)))$$
$$\forall n, a.(\mathsf{Av}(n, a) \leftrightarrow (\exists s.\mathsf{Emp}(n, s, a)))$$
$$\forall n.(\mathsf{Nv}(n) \leftrightarrow \mathsf{Mgr}(n))$$

(b) Physical design: $\Sigma_{\mathsf{LogPhys}}$

Fig. 2. An employee database.

4. We propose practical heuristics, based on the results of the conditional tableau, that allow us to consider only *reasonable* plans, e.g., plans that do not contain irrelevant symbols, duplicate identical conjuncts, and so on.

In addition, while the proposed method is complete, that is, can enumerate all valid plans by brute force, our latest work on prototypes demonstrates that the heuristics introduced in the last part of the paper enable implementations with an efficiency comparable to existing RDBMS query optimizers. We found with work on earlier prototypes that alternative approaches based on generating variant interpolants by *backtracking* a standard analytic tableau are inexorably limited by the *subformula property* of tableau proofs. In particular, the space of query plans that can be constructed directly as interpolants extracted from closed tableau is severely limited, often not even covering all plans considered by standard RDBMSs. We also found that approaches based on backtracking of (tableau) proofs suffer from performance issues and, in practice, seem to work only for toy examples. The separation of a tableau reasoning phase from plan search proposed on this paper solves both of the these issues.

Last, as a side effect of restricting our attention to *range restricted* formulae and the associated *absorption normal form* developed in this paper, the tableau calculus can be considerably simplified, in particular in the way first-order variables are handled. The approach is a generalization of a similar idea developed for the clausal setting [17].

The following example illustrates the overall capabilities of our framework.

Example 1. Consider an application that manages information about employees $\mathsf{Emp}(n, s, a)$ in which n-values identify employees by serving as primary keys of Emp-tuples, and where s-values and a-values encode an employee's salary and age. Each employee n is also either a worker, $\mathsf{Wkr}(n)$, or a manager, $\mathsf{Mgr}(n)$, but never both. A logical design Σ_{Log} for this application is given by the FOL theory in Figure 2(a) in which the underlying signature Log consists of the set of predicate symbols $\{\mathsf{Emp}, \mathsf{Mgr}, \mathsf{Wkr}\}$. Observe that the first sentence is a functional dependency that ensures Emp-tuples are uniquely determined by n-values.

Now assume that a physical design of the data is given by a collection of three *materialized views* that constitute an integration of legacy data via the logical design [14]. In our framework, this is captured by augmenting the set of logical predicate symbols Log with a set of three additional physical predicate symbols $\mathsf{Phys} = \{\mathsf{Sv}, \mathsf{Av}, \mathsf{Nv}\}$ abstracting the material capabilities of three data sources (Sv and Av can be considered a *column-store* style representation of employees), and by augmenting the logical theory Σ_{Log} with the sentences $\Sigma_{\mathsf{LogPhys}}$ in Figure 2(b).

A user query that "finds distinct salary and age pairs for which some worker has the combination of the salary and age" (in preparation, say, for some big-data analysis) can be formulated over Log as follows:

$$\{(s, a) \mid \exists n.(\mathsf{Emp}(n, s, a) \wedge \mathsf{Wkr}(n))\}.$$

Together with the theory Σ defined in Figure 2, our framework is able to find the following query plan formulated over Phys for computing this query:

$$\{(s, a) \mid \exists n.(\mathsf{Sv}(n, s) \wedge \mathsf{Av}(n, a) \wedge \neg\mathsf{Nv}(n))\}.$$

The query plan is then transformed to executable code by substituting access paths for atomic subformulae and providing (often quite straightforward) templates implementing the necessary logical operations [23]. Note that all sentences comprising Σ are required for this derivation, including the functional dependency on Emp. $\square$

Note that for realistic examples our experimental implementation executes hundreds of inferences and generates numerous variant query plans. Hence, due to space limitations, it will be necessary to use the above simple (and slightly artificial example) to illustrate facets of how plans are found in our framework. For much more complex examples see [23].

The paper is organized as follows. After a review of related work, Section 2 provides the necessary background definitions and review. Our main results are then given in Section 3, beginning with a review of an absorption procedure that our framework applies to a given physical design Σ. With the assumption that a user query or the sentences occurring in a physical design are range-restricted, the procedure ensures the possibility for tableau in which free variables do not occur. The remaining subsections introduce our framework in which we define conditional tableau, and in which we define a property linking a conditional tableau with a query plan. Note that this property can be easily computed and holds if and only if the query plan is equivalent to some interpolant generated by some closed tableau over Σ and query Q. The final subsection considers more practical issues in plan search in which this property can be used to guide the search for query plans in a bottom-up manner. In Section 4, we discuss issues beyond the scope of the paper but that must be addressed in any practical setting.

1.1 Review of related work

Query optimization has been a focus of research since the introduction of the relational model by Codd [7]. However, the approach currently adopted by most RDBMSs assumes what might be called a *standard design* in which there is always an efficient way of finding a plan for any user query of the form $P(\overline{x})$, where P denotes any predicate over which quantification is permitted, e.g., by assuming a *base file* and possibly several associated *search indices*. In such cases, finding an acceptable query plan typically reduces to a cost-based optimization problem [6, 13]. However, there are important circumstances in which such an assumption is no longer valid. The problem of integrating information from multiple legacy sources and logical designs that manifest a more sophisticated and independently developed ontological appreciation of an application domain are important examples. To address this issue, approaches to rewriting user queries in terms of materialized views (i.e., cached results of earlier user queries) were developed [15]. Subsequently, Deutsch et al. [9] have developed an approach to finding plans under constraints formulated as more general forms of FOL dependencies, also based on the *chase* [2, 16].

For first-order queries and constraints, the work of Beth [4] on implicit definability yields a complete criterion for the existence of a query plan and *Craig interpolation* [8] combined with an appropriate proof system, such as the *sequent calculus* [12], can be the basis of a method to synthesize query plans. Note that interpolation-based techniques are complete if arbitrary interpretations are considered. If only finite interpretations are allowed, the methods remain sound, albeit incomplete [10]. Beth definability and interpolation have been considered for fragments of FOL characterized by description logics [21, 22] and for the guarded fragment of FOL [18]. Our work builds on earlier work that, to the best of our knowledge, was the first to propose a general framework for finding query rewritings in FOL and in which a logical design can be defined by arbitrary domain independent formulae [5]. Refinements and extensive application scenarios were presented in [23]. Subsequently, an interpolation-based approach to query rewriting that accounts for binding patterns within the proof system itself has been proposed in [3]. However, this approach appears to suffer from the above-mentioned problems related to backtracking and the subformula property of tableau proofs.

2 Background and Definitions

We adopt standard definitions of databases, queries, and query answers [1]. This means that database instances are identified with *first-order* interpretations of non-logical symbols, and that query answers are relational instances consisting of tuples of domain elements that, when used as a finite valuation of the query's free variables, make the query true in a given (database) interpretation. We also assume that FOL formulae that are user queries or that occur in logical and physical designs satisfy the following syntactic restriction.

Definition 1 (Range-restricted First-order Formulae). *Let S be a set of predicate symbols. The set of* range-restricted *S-formulae is defined by the following grammar*

$$Q, Q' \ ::= \ R(\bar{x}) \ \mid \ Q \wedge Q' \ \mid \ Q \wedge (x = y) \ \mid \ \exists x.Q \ \mid \ Q \wedge \neg Q' \ \mid \ Q \vee Q',$$

where $R \in S$ and where the last four cases satisfy the following respective conditions: $y \in \mathsf{Fv}(Q)$, $x \in \mathsf{Fv}(Q)$, $\mathsf{Fv}(Q') \subseteq \mathsf{Fv}(Q)$ and $\mathsf{Fv}(Q') = \mathsf{Fv}(Q)$ (union compatibility).

Note that this still includes the standard relational algebra, the first-order fragment of the SQL query language, as well as a wide variety of database constraints including so-called algebraic dependencies [1]. Our goal can now be characterized in terms of *query rewriting*: a search for an equivalent formula that only contains a subset of the non-logical symbols, those symbols that represent the actual data sources such as materialized views, and therefore range-restricted queries containing those symbols can be *executed* [23].

Definition 2 (Query Rewriting under Constraints). *Let* Log *and* Phys *denote sets of predicate symbols, Σ be a set of closed* $(\mathsf{Log} \cup \mathsf{Phys})$*-formulae and Q a* Log*-formula. A* rewriting *of Q over* Phys *and under Σ is a* Phys*-formula P such that $\Sigma \models Q \leftrightarrow P$.*

In the following, we denote a query rewriting problem as a triple $(Q, \Sigma, \mathsf{Phys})$. Also note that a rewriting may not exist. This happens if Phys does not provide sufficient material capabilities to determine an answer to Q.

Example 2. The requirement that Σ only contains range-restricted formulae is necessary. In particular, consider the physical design given by

$$\Sigma = \{\forall x.P(x) \vee R(x), \forall x.\neg P(x) \vee \neg R(x)\},$$

with $\mathsf{Log} = \{P\}$ and $\mathsf{Phys} = \{R\}$. Then the (only) rewriting of the user query $P(x)$ is $\neg R(x)$, a formula that is not range-restricted nor equivalent to a range restricted formula.

On Interpolation. Although the constructions in the remaining parts of the paper are self-contained, the proof outlines refer, on several occasions, to a tableau-based interpolation technique outlined in detail in [11]. This technique relies on (1) introducing *biased* formulae labeled by L and R, a notation that is reused in the remainder of this paper, on (2) extending the tableau rules to the biased formulae, and on (3) *adorning* these rules with *interpolant extraction* annotations. For example, for two of the base cases of tableau *clashes*, the extraction is defined as follows:

$$S \cup \{A^L, \neg A^L\} \to \bot \text{ and } S \cup \{A^L, \neg A^R\} \to A.$$

The first case is called an "L-L" clash and the second case an "L-R" clash. Also note the *interpolant* is located after the arrow in these rules. This notation is extended to all inference rules, e.g.,

$$\frac{S \cup \{\varphi_1^R\} \to P_1 \qquad S \cup \{\varphi_2^R\} \to P_2}{S \cup \{(\varphi_1 \vee \varphi_2)^R\} \to P_1 \wedge P_2} \ (\mathsf{disj\text{-}R}),$$

and, in this way, complex interpolants are generated. For details please see [11].

3 Enumerating Rewritings via Tableau

To generate alternative query rewritings, we employ a modified variant of interpolant generation based on tableau refutation proofs [11]: given a query rewriting problem $(Q, \Sigma, \mathsf{Phys})$, the Craig theorem [8] can be used to show than a rewriting exists if and only if

$$\Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models Q^L \to Q^R,$$

where Σ^L (resp. Σ^R) is the set of sentences in which every occurrence of a non-logical symbol P has been replaced by P^L (resp. P^R), and where

$$\Sigma^{LR} = \{\forall \bar{x}.P^L(\bar{x}) \leftrightarrow P(\bar{x}), \forall \bar{x}.P^R(\bar{x}) \leftrightarrow P(\bar{x}) \mid P \in \mathsf{Phys}, \bar{x} = \mathsf{Fv}(P)\}.$$

Q^L (Q^R) is the user query in which non-logical symbols are renamed analogously. This formulation will be presented to a tableau reasoner in a refutation form: formulas δ over the physical schema Phys such that

$$\Sigma^L \cup \Sigma^R \cup \Sigma^{LR} \models Q^L \to \delta \to Q^R.$$

It is immediate that δ is then equivalent to Q under Σ, as required.

Note that the standard approach to interpolation [11] has a simpler definition of the logical implication problem in which non-logical symbols in Phys are not renamed in either Σ^L or Σ^R. Our refinement is a crucial first step in factoring tableau reasoning shared by alternative query plans for a user query, and is essential to the notion of a conditional tableau introduced below.

3.1 Absorption

We take *absorption* as the goal of ensuring that all universal subformulae are of the form "$\forall \bar{x}.(R(\bar{x}) \to Q)$". Note that Q can be another absorbed formula to facilitate multiple premises in the absorbed formulae. The utility of absorption in our method is twofold.

1. In a fashion similar to standard absorptions, unless a positive (ground) instance of R is derived, there is no requirement to explore the above implication, in particular potential disjunctions in the Q subformula.
2. Also, the normal form defined below guarantees that the instantiation of universally quantified variables $\bar{x}$ is completely determined by a matching positive ground instance of R.

Definition 3 (Absorption Normal Form (ANF)). *Formulae in* absorption normal form *are given by the grammar*

$$Q \;::=\; R(\bar{t}) \;\mid\; \bot \;\mid\; Q \wedge Q \;\mid\; Q \vee Q \;\mid\; \forall \bar{x}.R(\bar{x}, \bar{t}) \to Q[\bar{s}/\bar{x}],$$

where R ranges over a set of predicate symbols, $\bar{x}$ is a tuple of variables, $\bar{t}$ and $\bar{s}$ tuples of ground terms, and $[\bar{s}/\bar{x}]$ a substitution that replaces all s-terms with x-variables.[2]

In addition, all *existential* quantifiers are Skolemized in a standard way. There is no real loss of generality in requiring the input to the tableau proof procedure to be in ANF. In particular, with the exception of the identically true sentence $\top$ (which serves no useful purpose as either a constraint or a query), every range-restricted formula can be *compiled* to an ANF formula and, therefore, so can every range-restricted sentence. We have developed an algorithm that transforms range-restricted formulae to ANF; the details are straightforward and are omitted due to space limitations. Hence, without loss of generality, we assume hereon that all formulae input in our reasoning procedures are in ANF.

Example 3. Consider two range restricted sentences $\forall n.(\mathsf{Mgr}(n) \rightarrow \neg\mathsf{Wkr}(n))$ and $\forall n.((\exists s, a.\mathsf{Emp}(n, s, a)) \leftrightarrow (\mathsf{Mgr}(n) \vee \mathsf{Wkr}(n)))$ from our introductory example. The respective ANF transformations are given as the sentence $\forall n.(\mathsf{Mgr}(n) \rightarrow (\mathsf{Wkr}(n) \rightarrow \bot))$ for the former, and as the three sentences $\forall n, s, a.\mathsf{Emp}(n, s, a)) \rightarrow (\mathsf{Mgr}(n) \vee \mathsf{Wkr}(n)))$, $\forall n.(\mathsf{Mgr}(n) \rightarrow \mathsf{Emp}(n, \mathsf{sk}_s(n), \mathsf{sk}_a(n))$ and $\forall n.(\mathsf{Wkr}(n) \rightarrow \mathsf{Emp}(n, \mathsf{sk}_s(n), \mathsf{sk}_a(n))$ for the latter.

3.2 Conditional Tableau

We now introduce the above-mentioned *conditional tableau*, a mechanism that enables us to reason about a physical design Σ and the user query Q in a way that supports subsequent reasoning-free plan generation.

Definition 4 (Conditional Formulae). *Let* Phys *be a set of (physical) predicate symbols, φ a formula in ANF and C a set of ground atoms over* Phys*. A* conditional formula *is an expression of the form $\varphi[C]$. We call C a condition (for φ).*

We consider all ANF formulae to also be conditional formulae with an empty set of conditions, denoted $\varphi[\,]$. The conditional formulae allow us to define conditional tableau that are an extension of analytic tableau in which parts of the inferences are marked to be optional and therefore dependent on whether a particular physical predicate is used in the ultimate query plan. In this way, the conditional tableau facilitates schema reasoning for *all* query plans *without* committing to a particular choice of physical predicates, e.g., the choice of which index to use for a particular relation (and the subsequent need for backtracking to find alternatives).

Definition 5 (Conditional Tableau Inference Rules). *Let S be a set of conditional formulae and* Phys *a set of (physical) predicate symbols. We build*

[2] We assume that equality is simply an additional predicate symbol constrained by the standard axioms. A more efficient, paramodulation-style treatment of equality is an orthogonal issue and is beyond the scope of this paper.

a conditional tableau proof tree T *for* S *and* Phys *by applying the following* inference rules *(presented as proof rules):*

$$\frac{S \cup \{\varphi[C], \psi[C]\}}{(\varphi \wedge \psi)[C] \in S}\ (\mathsf{conj}) \qquad \frac{S \cup \{\varphi[C]\} \qquad S \cup \{\psi[C]\}}{(\varphi \vee \psi)[C] \in S}\ (\mathsf{disj})$$

$$\frac{S \cup \{(\varphi[\bar{t}/\bar{x}])[C \cup D]\}}{\{R(\bar{t})[C], (\forall \bar{x}.R(\bar{x}) \to \varphi)[D]\} \subseteq S}\ (\mathsf{abs}) \quad \frac{S \cup \{R(\bar{t})[R(\bar{t})]\}}{S}\ \ R \in \mathsf{Phys}\ (\mathsf{phys})$$

As usual, the inference rules are only applied when their consequent differs from the antecedent. Also, due to the fact that all formulae in the input to the problem are in ANF, no free variables will ever appear in the tableau. The inference rules yield the notion of conditional tableau as follows:

Definition 6 (Conditional Tableau for $(Q, \Sigma, \mathsf{Phys})$**).** *Let* $(Q, \Sigma, \mathsf{Phys})$ *be a rewriting problem. A* conditional tableau *for* $(Q, \Sigma, \mathsf{Phys})$ *is a pair of tableau proof trees* (T^L, T^R) *utilizing inference rules in Definition 5, where* T^L *is a proof tree for* $\Sigma^L \cup \{Q^L(\bar{a})\}$ *and* $\{P^L \mid P \in \mathsf{Phys}\}$ *and* T^R *is a proof tree for* $\Sigma^R \cup \{Q^R(\bar{a}) \to \bot\}$ *and* $\{P^R \mid P \in \mathsf{Phys}\}$ *for* $\bar{a}$ *a tuple of distinct constants replacing the free variables of* Q*.*

The conditional tableau proof tree *compactly abstracts* a family of standard analytic tableau proof trees (each of which can be selected for by choosing a set of conditional atoms).

Example 4. A conditional tableau for our introductory example will contain four branches, two in T^L and two in T^R as shown in Figure 3 (in the figure we reuse the variable names, perhaps primed, as Skolem terms to improve readability, i.e., $n = \mathsf{sk}_n(s, a)$, $a' = \mathsf{sk}_a(n, s)$, and $s' = \mathsf{sk}_s(n, a)$).

Note that, unless the rewriting problem is trivial (i.e., the query is unsatisfiable w.r.t. Σ), the conditional tableau will not be *closed in the standard sense.* It will, however, provide a guidance to constructing a closed tableau starting by a top-level tableau constructed solely by using the *physical formulae* Σ^{LR}, (see Definition 8 below) and then *appending* the appropriate proof tree selected from the conditional tableau by branches of the top-level tableau. The interface between the branches of the top-level tableau and the conditional tableau is formalized using the notion of *closing set*:

Definition 7 (Conditional Tableau Closure). *We say that a set of* Phys*-literals* S closes *a conditional tableau proof tree* T *if there is a mapping* θ *from terms in* T *to terms in* S *(extended to literals) such that,* $\theta(\bar{a}) = \bar{a}$*,* $\theta(t_1) \neq \theta(t_2)$ *whenever* $t_1 \neq t_2$ *and* t_1, t_2 *are introduced in the same branch, and for each branch of* T *either of the following holds.*

1. *There is an atom* $R(\bar{t})[C]$*,* $R(\bar{t}) \notin C$*, such that* $\theta(C) \cup \{\neg R(\theta(\bar{t}))\} \subseteq S$*.*
2. *There is* $\bot[C]$ *such that* $\theta(C) \subseteq S$*.*

We call S *a* closing set *for* T*.*

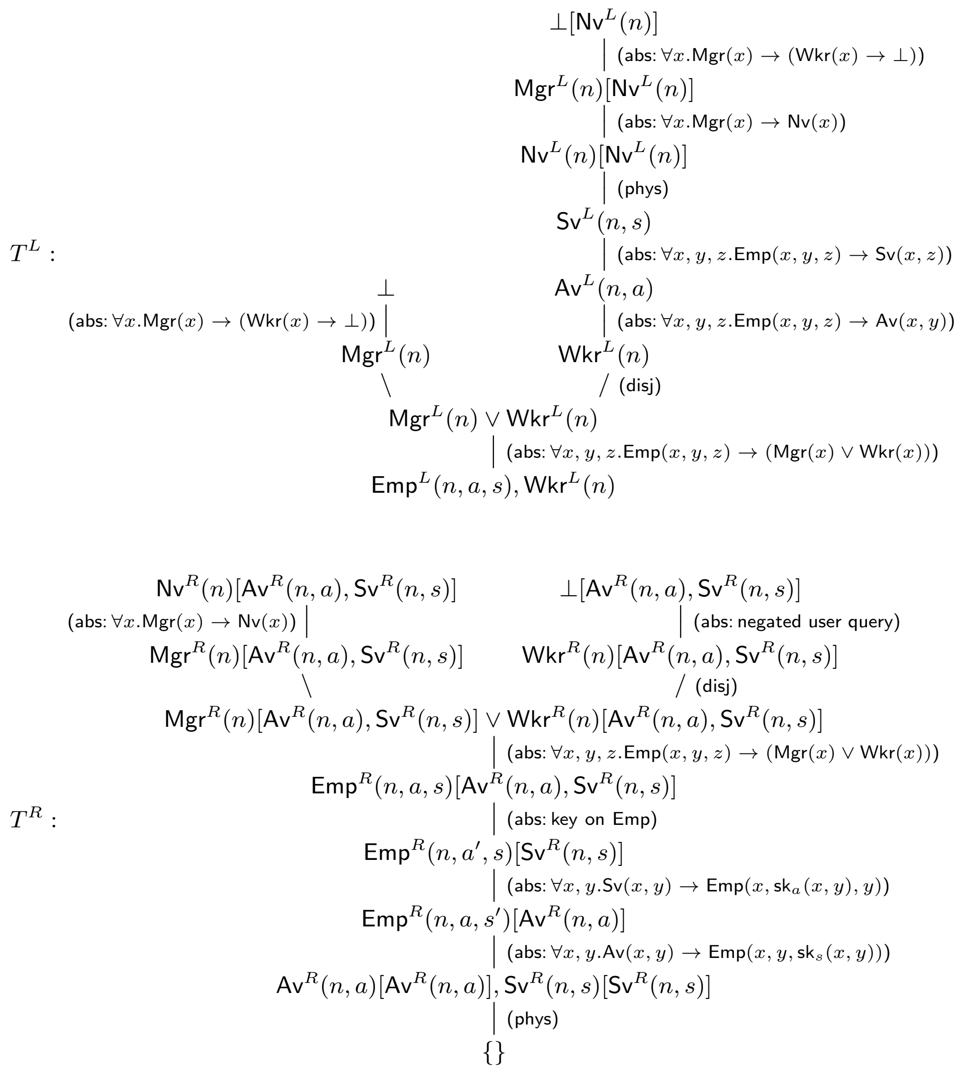
T^L :
$\bot[\mathsf{Nv}^L(n)]$
(abs: $\forall x.\mathsf{Mgr}(x) \rightarrow (\mathsf{Wkr}(x) \rightarrow \bot)$)
$\mathsf{Mgr}^L(n)[\mathsf{Nv}^L(n)]$
(abs: $\forall x.\mathsf{Mgr}(x) \rightarrow \mathsf{Nv}(x)$)
$\mathsf{Nv}^L(n)[\mathsf{Nv}^L(n)]$
(phys)
$\mathsf{Sv}^L(n,s)$
(abs: $\forall x,y,z.\mathsf{Emp}(x,y,z) \rightarrow \mathsf{Sv}(x,z)$)
$\bot$
$\mathsf{Av}^L(n,a)$
(abs: $\forall x.\mathsf{Mgr}(x) \rightarrow (\mathsf{Wkr}(x) \rightarrow \bot)$)
(abs: $\forall x,y,z.\mathsf{Emp}(x,y,z) \rightarrow \mathsf{Av}(x,y)$)
$\mathsf{Mgr}^L(n)$
$\mathsf{Wkr}^L(n)$
(disj)
$\mathsf{Mgr}^L(n) \vee \mathsf{Wkr}^L(n)$
(abs: $\forall x,y,z.\mathsf{Emp}(x,y,z) \rightarrow (\mathsf{Mgr}(x) \vee \mathsf{Wkr}(x))$)
$\mathsf{Emp}^L(n,a,s), \mathsf{Wkr}^L(n)$
$\mathsf{Nv}^R(n)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
$\bot[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
(abs: $\forall x.\mathsf{Mgr}(x) \rightarrow \mathsf{Nv}(x)$)
(abs: negated user query)
$\mathsf{Mgr}^R(n)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
$\mathsf{Wkr}^R(n)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
(disj)
$\mathsf{Mgr}^R(n)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)] \vee \mathsf{Wkr}^R(n)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
(abs: $\forall x,y,z.\mathsf{Emp}(x,y,z) \rightarrow (\mathsf{Mgr}(x) \vee \mathsf{Wkr}(x))$)
T^R :
$\mathsf{Emp}^R(n,a,s)[\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s)]$
(abs: key on Emp)
$\mathsf{Emp}^R(n,a',s)[\mathsf{Sv}^R(n,s)]$
(abs: $\forall x,y.\mathsf{Sv}(x,y) \rightarrow \mathsf{Emp}(x,\mathsf{sk}_a(x,y),y)$)
$\mathsf{Emp}^R(n,a,s')[\mathsf{Av}^R(n,a)]$
(abs: $\forall x,y.\mathsf{Av}(x,y) \rightarrow \mathsf{Emp}(x,y,\mathsf{sk}_s(x,y))$)
$\mathsf{Av}^R(n,a)[\mathsf{Av}^R(n,a)], \mathsf{Sv}^R(n,s)[\mathsf{Sv}^R(n,s)]$
(phys)
$\{\}$

Fig. 3. A CONDITIONAL TABLEAU.

Example 5. The closing sets in our example are $\{\neg\mathsf{Av}^L(n,a)\}$, $\{\neg\mathsf{Sv}^L(n,s)\}$, and $\{\mathsf{Nv}^L(n)\}$ for T^L and $\{\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,s), \neg\mathsf{Nv}^R(n)\}$ for T^R.

The mapping θ (not needed in our example) accounts for differences in choices of *Skolem terms* in independent branches during the construction of the conditional tableau: it is necessary to be able to construct a plan $\exists y.A(x,y) \vee B(x,y)$, assuming $\mathsf{Phys} = \{A, B\}$, for the query $(\exists y_1.A(x,y_1)) \vee (\exists y_2.B(x,y_2))$. Note that using θ in Definition 7 is sound since one *could* have chosen the same term to witness two different existential quantifiers in independent branches during the construction of the conditional tableau. It is an easy exercise using the definitions for interpolant extraction in [11] to verify the following.

Lemma 1. Let S be an arbitrary closing set for T^L (resp. T^R). Then the interpolants extracted from T^L (resp. T^R) are $\bot$ (resp. $\top$).

Proof. (sketch) All clashes in the analytic tableau corresponding to T^L (resp. T^R) w.r.t. S are "L-L" (resp. "R-R") or $\bot^L$ (resp. $\bot^R$) clashes by inspection; applying conjunctions, disjunctions, or quantifiers ultimately yields $\bot$ (resp. $\top$).

3.3 Query Plans

Now we complement the exploration of the *conditional tableau* with the actual plan enumeration phase. We first show how, given a query plan candidate, a range-restricted formula over Phys, we can test whether this plan is equivalent to the user query with respect to the database schema.

Definition 8 (Tableau for Query Plans). *Let P be a range-restricted formula over* Phys. *We inductively define two sets L_P and R_P on the structure of P, with each consisting in turn of sets of literals, as follows:*

P	:	L_P	R_P
$R(\bar{t})$	:	$\{\{\neg R^L(\bar{t})\}\}$	$\{\{R^R(\bar{t})\}\}$
$P_1 \wedge P_2$	:	$L_{P_1} \cup L_{P_2}$	$\{S_1 \cup S_2 \mid S_1 \in R_{P_1}, S_2 \in R_{P_2}\}$
$P_1 \vee P_2$	:	$\{S_1 \cup S_2 \mid S_1 \in L_{P_1}, S_2 \in L_{P_2}\}$	$R_{P_1} \cup R_{P_2}$
$\neg P_1$	:	$\{\{L^L(\bar{t}) \mid L^R(\bar{t}) \in S\} \mid S \in R_{P_1}\}$	$\{\{L^R(\bar{t}) \mid L^L(\bar{t}) \in S\} \mid S \in L_{P_1}\}$
$\exists x.P_1$	:	$L_{P_1[t/x]}$	$R_{P_1[t/x]}$

where t is an appropriate Skolem term, $L(\bar{t})$ denotes a literal, and $R(\bar{t})$ is an atom, both over Phys.

The construction of the fragments P in the above definition must also adhere to the restrictions imposed on *range-restricted* formulae such as *union compatibility.* The definition short-circuits an explicit construction of a tableau proof that yields the sought-after plan (or an equivalent formula) by defining the sets of literals that will be present on open branches of such a tableau. The following Lemma shows soundness of this construction.

Lemma 2. Let P be a range-restricted formula over Phys. Then there is an analytic tableau tree T^P that uses only formulae in $\Sigma^{LR} \cup \{\forall x.\mathsf{true}^R(x)\}$ such that the following holds.

1. Each open branch of T^P contains all literals in a set $S \in L_P \cup R_P$ (when $S \in L_P$, we call such a branch a *left branch*; otherwise, when $S \in R_P$, we call such a branch a *right branch*); and
2. The interpolant extracted from this tableau, assuming that further extensions of all left (right) branches interpolate to $\bot$ ($\top$, resp.), is logically equivalent to P.

Proof. (sketch) By case analysis. The base case $R(\bar{t})$ follows immediately from expanding the formulae $\forall \bar{x}.R^L(\bar{x}) \to R(\bar{x})$ and $\forall \bar{x}.R(\bar{x}) \to R^R(\bar{x})$ from Σ^{LR}, thus yielding an "L-R" clash on $R(\bar{t})$ and two open branches $\{\{\neg R^L(\bar{x})\}\}$ and $\{\{R^R(\bar{x})\}\}$.

In the case of a conjunction (resp. disjunction) $P_1 \wedge P_2$ (resp. $P_1 \vee P_2$), we simply attach the tableau for P_1 to all open right (resp. left) branches of the tableau for P_2; it is then a trivial but tedious exercise to verify that the claims hold.

In the case of negation $\neg P_1$, the claim holds by observing that applying a NNF-like procedure, that essentially replaces all L formulae by R formulae and vice versa, and using the reverse implications in Σ^{LR} to those used in the base case yields the desired result.

Finally, for existential quantification, we expand the auxiliary tautology $\forall x.\mathsf{true}^R(x)$ using an appropriate term t in the root of the tableau for $P_1[t/x]$. This arrangement *reinstates* the quantifier when the interpolant is extracted as required by the Lemma. Note that applying the negation case to this construction changes the $\mathsf{true}^R(x)$ into $\mathsf{true}^L(x)$, which in turn yields a universal quantifier in the interpolant, as expected.

To link the conditional tableau (T^L, T^R) with the descriptions (L_P, R_P) of open branches of the plan tableau T^P we use the following definition.

Definition 9. *We say that* (L_P, R_P) closes (T^L, T^R) *if* S *closes* T^L *for all* $S \in L_P$ *and* S *closes* T^R *for all* $S \in R_P$*, where* (T^L, T^R) *is the conditional tableau for* $(Q, \Sigma, \mathsf{Phys})$*.*

We illustrate the construction by appeal to our running example:

Example 6. For the desired plan, $P = \{(s,a) \mid \exists n.(\mathsf{Sv}(n,s) \wedge \mathsf{Av}(n,a) \wedge \neg \mathsf{Nv}(n))\}$, the sets are as follows:

$$L_P = \{\{\neg \mathsf{Av}^L(n,a)\}, \{\neg \mathsf{Sv}^L(n,a)\}, \{\mathsf{Nv}^L(n)\}\},$$
$$R_P = \{\{\mathsf{Av}^R(n,a), \mathsf{Sv}^R(n,a), \neg \mathsf{Nv}^R(n)\}\}.$$

Note that these *match* the closing sets of the conditional tableau in Example 5.

Combining conditional tableau with the query plan tableau and utilizing the results of Lemmas 1 and 2 yields the following.

Theorem 1. *Let* $(Q, \Sigma, \mathsf{Phys})$ *be a rewriting problem and* P *a query plan over* Phys*. Then* P *is a plan for* Q *if* (L_P, R_P) *closes* (T^L, T^R)*.*

Proof. (sketch) The tableau T_P constructed for P using Lemma 2 with its open branches extended by T^L and T^R is closed (in the standard sense, cf. Lemma 1) and hence, again by Lemma 2, yields an interpolant equivalent to P.

Since the problem of finding rewritings for first-order queries is recursively enumerable (but not recursive), the above theorem yields a method for finding rewritings using brute force.

Theorem 2 (Completeness). *Let* $(Q, \Sigma, \mathsf{Phys})$ *be a rewriting problem and* P *a query plan for* Q*. Then there is a conditional tableau* (T^L, T^R) *that is closed by* (L_P, R_P)*.*

Proof. (sketch) Failure to find such a conditional tableau yields a saturated (infinite) proof tree from which we can extract a witness interpretation that satisfies Σ and $Q \wedge \neg P$ or $P \wedge \neg Q$, depending on whether the *failure* occurs in a left or in a right branch of T^P.

3.4 Practical Plan Search

We have shown how all plans can be enumerated by a brute force use of conditional tableau. However, not all plans merit any effort, for example, plans with sub-expressions of the form "$A(x) \wedge A(x)$", "$A(x) \vee \neg A(x)$", etc. We now show how, with conditional tableau, subsequent plan enumeration can be restricted in a way that can focus the search for query plans in a bottom-up manner that is based on the computation of minimal closing sets.

Controlled Conditional Tableau. It is necessary to first regulate the use of the (phys) inference rule in the conditional tableau proof trees. In particular, observe that, when a single physical atom $R \in \mathsf{Phys}$ is used in a hypothetical plan, on one hand, it will *close* branches containing R (in T^L), and on the other hand, it will allow exploring the conditional parts of the T^R tableau that are conditioned on R. Otherwise, the conditional parts will remain unexplored since, technically, they are not part of the actual tableau. The reverse is true when $\neg R$ is used in such a plan. Hence, one can restrict the use of the (phys) rule as follows:

$$\frac{S \cup \{R^L(\bar{t})[R^L(\bar{t})]\}}{S} \; R^R(\bar{t})[C] \in T^R \qquad \frac{S \cup \{R^R(\bar{t})[R^R(\bar{t})]\}}{S} \; R^L(\bar{t})[C] \in T^L$$

where $R(\bar{t})[C] \in T$ stands for "$R(\bar{t})[C]$ appears in some branch of T". (Indeed, the conditional tableau illustrated in Figure 3 is also controlled in this way.)

With this restriction, it becomes possible to *scan* the conditional tableau for (alternative) minimal closing sets for the conditional tableau (T^L, T^R). This can be done by inspecting the branches of the tableau for individual (conditional) atoms and $\perp$, each of which closes a particular branch, and then by computing minimal closing sets over these sets of atoms.

Bottom-up Plan Search. The *minimal closing sets* then guide the bottom-up search for query plans. Intuitively, we restrict the application of a bottom-up construction of query plans by comparing the sets L_P and R_P from Definition 8 with the minimal closing sets constructed from the conditional tableau. This allows us to eliminate plan fragments that cannot ultimately lead to closing the overall tableau, as required in Definition 9. This idea is formalized as follows.

Definition 10 (Relevant Plan Fragment). *Let H be a set of minimal closing sets for the conditional tableau (T^L, T^R). We say that a plan P is a* relevant plan fragment (for (T^L, T^R)) *if either of the following holds:*

1. *For all $S \in L_P \cup R_P$, there is $S' \in H$ such that $S \subseteq S'$; or*
2. *For all $S \in L_{\neg P} \cup R_{\neg P}$, there is $S' \in H$ such that $S \subseteq S'$.*

In addition, $\exists x.P[x/t]$ is only relevant if, whenever t appears in P and also in a literal in S and $S \subseteq S'$ as above, then t does not appear in any literal in $S' - S$.

The second item in the above definition allows for plans that are not in negation normal form and the last condition prevents incorrect application of quantifiers (applications that would, e.g., break apart join variables). This definition naturally constrains the bottom-up construction of plan fragments by requiring the resulting fragment be relevant and that its L_P and R_P sets are distinct from all its constituent sub-fragments.

Example 7. The relevant plan fragments for our example plan are $\mathsf{Sv}(n,s)$, $\mathsf{Av}(n,a)$, $\mathsf{Nv}(n)$, $\neg\mathsf{Nv}(n)$, $\mathsf{Sv}(n,s) \wedge \mathsf{Av}(n,a)$, $\mathsf{Sv}(n,s) \wedge \mathsf{Av}(n,a) \wedge \neg\mathsf{Nv}(n)$, $\exists x.\mathsf{Sv}(x,s) \wedge \mathsf{Av}(x,a) \wedge \neg\mathsf{Nv}(x)$, and so on. Examples of plan fragments that are *not* relevant are $\mathsf{Av}(n,a) \vee \mathsf{Sv}(n,s)$ and $\neg(\mathsf{Av}(n,a) \wedge \mathsf{Nv}(n))$.

Note also that restricting plans to relevant fragments naturally eliminates subformulae constructed from irrelevant symbols, such as tautologies $A \vee \neg A$ constructed from symbols not appearing in the given rewriting problem (but that must be allowed in Definition 9 for the completeness argument to hold). In addition, when plans are constructed bottom-up from fragments, additional heuristics can be used to eliminate unwanted plans, for example,

– when fragments are combined by a conjunction, we disallow combining fragments that contain the same sub-conjunct (such as $A \wedge B$ with $B \wedge C$);
– when fragments are combined by a disjunction, we follow a similar heuristic; moreover, we also allow to equate Skolem terms that belong to different branches of the conditional tableau to allow sharing variables and quantifiers (yielding an efficient way to realize the mapping θ in Definition 7);
– terms equated to other terms in the disjunctions above or replaced by quantified variables become *forbidden* and further conjunctions with fragments containing such terms are also disallowed.

The above heuristics allow efficient construction of query plans from fragments and are conditioned solely by the fragments' L_P and R_P sets and on how they compare to the minimal closing sets for the conditional tableau.

Overall, these arrangements allow us to use numerous plan search algorithms to construct relevant plan fragments ranging from a bottom-up dynamic programming style algorithms [20] to A^* based planning [19], all the while utilizing cost-based pruning for fragments with the same L_P, R_P sets.

4 Summary Comments

There are a number of issues relating to query plans beyond the framework outlined in this paper that must still be addressed in any practical setting, and that also make it necessary to refine how query plans are characterized.

1. First, so-called *binding patterns* are often needed in subplans. This happens, for example, in web-based information sources that disallow any “get all” client request for non-logical parameters mentioned in a plan due to the high costs of such requests. Integrating binding patterns (see [23] for formal definitions) to the proposed framework requires relaxing the definition of *relevant patterns* (Definition 10) to allow additional atomic fragments that provide *bindings* required by the plan fragments actually needed to close the tableau.
2. Second, an *iterator* or *bag semantics* is necessary to address the computational overhead of checking for duplicates computed by subplans. Indeed, our introductory example illustrates this requirement.
3. Third, the issue of variable typing in both user queries and query plans must be addressed in the ultimate code generated for a query plan in order to interface with host language code.
4. And fourth, there also remains a need to compare different query plans in terms of their execution times according to a cost model. Indeed, it is this cost model that ultimately drives the plan search phase outlined in Section 3.4.

These issues have been addressed for a plan language that refines the class of range-restricted FOL formulas by Toman and Weddell [23]. Notably, the authors introduce syntactic notions of *input variables* and *output variables* for formulae that address the above issues relating to binding patterns and bag semantics, and that also yield a number of additional benefits. First, logical operators occurring in plans are given a procedural interpretation. Conjunction is interpreted as *nested loops join* for example, and disjunction as *concatenation.* And second, one can introduce additional syntax to perform *cuts* and *duplicate elimination.* Both the refined notion of query plans in [23] and extra-logical notions of cost can be easily adapted to operate in an incremental fashion suited to the above approach to plan search.

We have implemented our proposed framework in our latest prototype that also addresses the issues outlined above. To reiterate our introductory comments, we found that employing the absorption technique is absolutely necessary for performance reasons. Otherwise, solving even the simplest rewriting problems become computationally infeasible. Recall that we also found that standard tableau interpolation techniques inexorably reduce the space of query plans for a given query, even with the refinement of having separate physical rules as outlined at the start of Section 3, and that a complete procedure for finding all possible plans almost certainly requires the level of indirection between interpolants and query plans manifest in Definitions 9 and 10.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley, 1995.
2. Alfred V. Aho, Catriel Beeri, and Jeffrey D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4:297–314, September 1979.

3. Michael Benedikt, Balder ten Cate, and Efthymia Tsamoura. Generating low-cost plans from proofs. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 200–211, 2014.
4. Evert Willem Beth. On Padoa's method in the theory of definition. *Indagationes Mathematicae*, 15:330–339, 1953.
5. Alexander Borgida, Jos de Bruijn, Enrico Franconi, Inanç Seylan, Umberto Straccia, David Toman, and Grant E. Weddell. On finding query rewritings under expressive constraints. In *SEBD*, pages 426–437, 2010.
6. Surajit Chaudhuri. An overview of query optimization in relational systems. In *PODS*, pages 34–43, 1998.
7. E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13:377–387, June 1970.
8. William Craig. Three uses of the Herbrand-Genzen theorem in relating model theory and proof theory. *Journal of Symbolic Logic*, 22:269–285, 1957.
9. Alin Deutsch, Lucian Popa, and Val Tannen. Physical data independence, constraints, and optimization with universal plans. In *Proc. International Conference on Very Large Data Bases*, VLDB '99, pages 459–470, 1999.
10. Heinz-Dieter Ebbinghaus and Jörg Flum. *Finite model theory (2nd ed.)*. Perspectives in Mathematical Logic. Springer, 1999.
11. Melvin Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition*. Graduate Texts in Computer Science. Springer Publishers, 1996.
12. Gerhard Gentzen. Untersuchungen über das logische schließen. I. *Mathematische Zeitschrift*, 39:176–210, 1935. 10.1007/BF01201353.
13. Yannis E. Ioannidis. Query optimization. *ACM Comput. Surv.*, 28(1):121–123, 1996.
14. Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
15. Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *PODS*, pages 95–104, 1995.
16. David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4:455–469, December 1979.
17. Rainer Manthey and François Bry. A hyperresolution-based proof procedure and its implementation in prolog. In *GWAI*, pages 221–230, 1987.
18. Maarten Marx. Queries determined by views: pack your views. In *PODS*, pages 23–30, 2007.
19. Nathan Robinson, Sheila A. McIlraith, and David Toman. Cost-based query optimization via AI planning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2344–2351, 2014.
20. Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*, pages 23–34, 1979.
21. Inanç Seylan, Enrico Franconi, and Jos de Bruijn. Effective query rewriting with ontologies over dboxes. In *IJCAI*, pages 923–925, 2009.
22. Balder ten Cate, Enrico Franconi, and Inanç Seylan. Beth definability in expressive description logics. In *IJCAI*, pages 1099–1106, 2011.
23. David Toman and Grant E. Weddell. *Fundamentals of Physical Design and Query Compilation*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.