Enumerating Query Plans via Conditional Tableau Interpolation

by

Eva Hu Feng

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2023

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

- Chapter 2 contains a summary of previous results and is co-authored with David Toman, with the exception of Section 2.2. Examples are modified from materials in [35].
- Chapter 3 contains a summary of previous results on free-variable tableau as well as new results authored by myself and aided by discussions with David Toman.
- Chapter 4 contains new proofs of existing results as well as joint work with David Toman, which has resulted in a conference paper [18].

Abstract

Database query optimization studies the problem of finding equivalent and efficient query execution plans for user queries under schema constraints. Logic-based approaches to query optimization leverage automated theorem proving and Craig interpolation to enumerate query plans that are correct and performance-optimal. In this thesis, we investigate and improve one of the state-of-the-art logic-based query optimizers – the Interpolation Test Bed (ITB).

We begin by formally capturing the physical data independence framework and query optimization problem with first-order logic. Then, we give a gentle introduction to the classical results from logic that form the basis of logic-based query optimizers.

We re-establish the correctness of ITB's conditional tableau interpolation mechanism by reduction to free-variable tableau interpolation. To facilitate the reduction proof, we introduce interpolation rules for the free-variable tableau and prove the correctness of interpolation. Then we show the correctness of conditional tableau interpolation by reduction.

We investigate a limitation of ITB's forward chaining design, which causes missing optimal plans. To address this limitation, we propose a rewriting procedure inspired by Magic Set Transformation (MST), to extend the plan space for the current ITB system. We show that the propose rewriting procedure effectively generates the missing query plans, which are otherwise not found, while accommodating the existing forward chaining design.

Acknowledgements

First and foremost, I would like to thank my supervisor, Prof. David Toman. I am deeply indebted to him for introducing me to logic and query optimization and offering invaluable technical insights and advice on research throughout my undergraduate and master's studies. His patience and encouragement gave me the courage to pursue research and overcome the challenges along the journey. This thesis would not have been possible without his support and guidance.

My sincere gratitude goes to my co-supervisor, Prof. Grant Weddell, for his feedback, advice and continuous interest in my work. Had he not introduced me to the world of databases and offer me my first research assistantship, my career trajectory would have taken a completely different course.

I would like to thank Prof. David Gosset for sparking my interest in combinatorics and optimization, especially graph theory, and supervising me as an undergraduate research assistant at IQC. I am grateful for his help and advice, which are crucial to starting my journey in research.

I appreciate Prof. Tamer Özsu and Prof. Richard Trefler for generously agreeing to read my thesis and giving helpful feedback.

Finally, I would like to thank my mother and father, S.J. Hu and X.F. Feng, for their unconditional love, endless patience, and constant support. Without their hard work and sacrifices, I would not have the education and opportunities I have today, especially this one.

Dedication

To my beloved mother and father.

Table of Contents

List of Figures

Chapter 1

Introduction

Any application that manages information, ranging from Excel to Google search, is powered and limited by its ability to store and retrieve data; in particular, its ability to answer queries and update data efficiently. A query plan implements how a high-level user query or update is answered through specifying a sequence of operations over the data storage. Performance among query plans, however, varies substantially [28, 16]. The study of query reformulation tackles the task of query optimization by finding logically equivalent and computationally efficient query plans for each user query or update [1].

Traditional query optimization techniques, such as join ordering based on cost models [20] and fixed transformations based on an initial plan, preserve logical equivalence without directly confronting the query reformulation problem [28, 16]. However, these transformations commonly explore only a restricted neighbourhood around the initial plan, limiting improvements to a local optimum [7].

Over the last decade, a novel logic-based approach has emerged to address the problem of query reformulation under constraints. Since the set of all equivalent query plans is recursively enumerable under unrestricted semantics, and logic-based approaches can, in principle, search over all possible query plans [7, 31, 25]. State-of-the-art logic-based query optimizers, Interpolation Test Bed (ITB) [32] and Proof Driven Querying (PDQ) [8], both leverage automated theorem proving techniques and Craig interpolation (Section 2.2) to produce logically equivalent query plans [32, 25, 31, 14]. In particular, a variant of the analytic tableau proof system, called the conditional tableau, is developed for ITB to construct Craig interpolants for the purpose of query optimization. The correctness of conditional tableau inference and interpolation has been established previously in [25]. Since then, the conditional tableau and ITB have undergone significant changes and improvements. The

aim of this thesis is twofold:

1. Establish the correctness of the conditional tableau in full details; and
2. Extend the space of enumerable query plans for the current ITB system.

For (1), we prove the correctness of conditional tableau interpolation by reduction to another tableau variant called the free-variable tableau [21]. To our knowledge, previous work has not explored interpolation in the context of free-variable tableau [21]. To facilitate the reduction proof, in Chapter 3, we propose a set of interpolation rules for the free-variable tableau and prove the correctness of free-variable tableau interpolation. Following the preparatory work on the free-variable tableau, in Chapter 4, we present the inference rules and interpolation mechanism for the conditional tableau and prove the correctness of conditional tableau interpolation.

For (2), we present a rewriting procedure inspired by the Magic Set Transformation (MST) technique originally developed in the area of logic programming [6, 5] in Section 4.3. The current ITB system implements the conditional tableau using forward chaining and derives positive ground literals only to improve the performance of theorem proving. However, a limitation of the current design is that certain negative ground literals can never be generated, which leads to missing potentially optimal query plans. We show that the proposed MST-based rewriting procedure addresses the aforementioned limitation by extending the space of query plans.

In addition to the conditional tableau, an independent planning component [29] based on the A^* search algorithm [22] is responsible for efficient query plan enumeration. We summarize ITB's system design and implementation details in Chapter 5.

Finally, we briefly comment on related work and future directions in Chapter 6.

Chapter 2

Background

2.1 Query Optimization

Abstraction is one of the fundamental principles in computer science. Two canonical examples are the idea of *data independence* [4, 3] from database systems and the idea of *abstract data type* (ADT) [27] from data structures – both can be dated back to the early 70s. Making the interface and implementation independent of each other provides numerous advantages in application development, including user-friendliness, ease of maintenance, better performance and scalability, greater flexibility, improved security, etc. For instance, modern file systems allow the users to manipulate files without understanding the low-level disk layouts and enable the developers to optimize disk layouts without affecting the user interface.

Today, data independence is a standard practice in the area of database design, following the development of the relational model (RM) [12] with accompanying data manipulation languages [13] based on first-order logic [17]. However, the abstraction of low-level implementation, such as C programs, in database management systems (DBMSs) often comes at the cost of increased development time and decreased quality of deployed systems. This thesis describes a novel approach [11, 31, 25, 32, 34] to tackle this crucial challenge inherent to data independence: automating the search for performance-optimal, low-level instructions that correctly implement high-level user commands in DBMSs.

The remainder of Section 2 describes the data independence property and query optimization problem for relational databases in terms of first-order signatures and integrity constraints (i.e., sentences over these signatures).

2.1.1 The Logical Schema

The *logical schema* captures the user-level conceptual description of a database. It consists of a first-order signature $\mathsf{S}_{\mathsf{Log}}$, called the *logical vocabulary*, and a set of integrity constraints Σ_{Log} over $\mathsf{S}_{\mathsf{Log}}$, called the *logical constraints*.

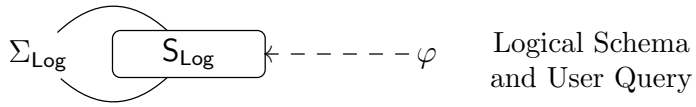


Figure 2.1: Logical schema.

Users can request data from the database by formulating a *user query* φ over the atoms in $\mathsf{S}_{\mathsf{Log}}$ under the constraints in Σ_{Log}. From an application perspective, $\mathsf{S}_{\mathsf{Log}}$ represents the set of all tables that are visible to the users, and user queries must be formulated using tables in $\mathsf{S}_{\mathsf{Log}}$ under the constraints imposed by Σ_{Log}. Notably, the following are sufficient for database users to develop applications:

1. Familiarity with $\mathsf{S}_{\mathsf{Log}}$ and Σ_{Log}; and
2. Presumption of a single database instance (i.e., a single interpretation on $\mathsf{S}_{\mathsf{Log}}$ that is a model of Σ_{Log}) instead of all valid instances.

Example 2.1 (Logical Schema) We illustrate the features of the logical schema through the following SQL declaration.

```
CREATE TABLE employee (
  num     INTEGER NOT NULL,
  name    CHAR(20),
  worksin INTEGER  NOT NULL
  PRIMARY KEY (num),
  FOREIGN KEY (worksin)
        REFERENCES department
)
```

```
CREATE TABLE department (
  num     INTEGER NOT NULL,
  name    CHAR(50),
  manager INTEGER NOT NULL,
  PRIMARY KEY (num),
  FOREIGN KEY (manager)
        REFERENCES employee
)
```

The above declarations specify the content of *employee* and *department* instances: *employee* contains information about employee numbers, names and departments they work in, and *department* contains information about departments' names and managers. In our

formalism, this is equivalent to the following signatures $\mathsf{S}_{\mathsf{Log}}$ (where "/i" indicates predicate arity) and integrity constraints Σ_{Log}:

$$\mathsf{S}_{\mathsf{Log}} = \{\texttt{employee}/3, \texttt{department}/3, \texttt{manager}/3, \texttt{worker}/3\},$$

$\Sigma_{\mathsf{Log}} = \{$

// All free variables are assumed to be universally quantified.

*// **employee** numbers are unique*
$\texttt{employee}(x, y_1, z_1) \wedge \texttt{employee}(x, y_2, z_2) \rightarrow y_1 = y_2 \wedge z_1 = z_2,$

*// every **employee** belongs to a **department***
$\texttt{employee}(x, y, z) \rightarrow \exists u, v.\texttt{department}(z, u, v),$

*// a (virtual) view for **manager**s*
$\texttt{manager}(x, y, z) \leftrightarrow (\texttt{employee}(x, y, z) \wedge \exists n.\texttt{department}(z, n, x))),$

*// disjoint partition of **employee**s to **manager**s and **worker**s*
$\texttt{employee}(x, y, z) \leftrightarrow (\texttt{manager}(x, y, z) \vee \texttt{worker}(x, y, z)),$
$\texttt{manager}(x, y, z) \rightarrow \neg\exists u, v.\texttt{worker}(x, u, v),$

*// business logic: **manager**s work for their own **department**s*
$(\texttt{department}(x, y, z) \wedge \texttt{employee}(z, u, w)) \rightarrow x = w$

$\}.$

As illustrated by Σ_{Log}, the user can specify integrity constraints in the logical schema to express advanced functionalities that are beyond the typical implementations of the relational model, such as view, business rule, and partition.

2.1.2 The Physical Schema

The *physical schema* captures the low-level implementation of a database and the means to retrieve data from physical storage, such as disks. The physical schema, similar to its logical analog, consists of a *physical vocabulary* $\mathsf{S}_{\mathsf{Phys}}$ and a set of *physical constraints* Σ_{Phys} over both the logical and physical vocabulary, $\mathsf{S}_{\mathsf{Log}} \cup \mathsf{S}_{\mathsf{Phys}}$. A subset of symbols S_{A} in the physical vocabulary $\mathsf{S}_{\mathsf{Phys}}$ corresponds to *access paths* which are subroutines that retrieve data from *data structures* in physical storage.

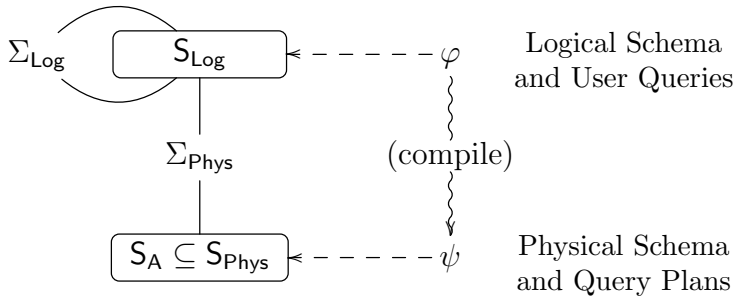


Figure 2.2: Logical and physical schema.

Each access path is associated with a *binding pattern* which specifies the input and output variables of the access path. Values for the input variables must be given to the access path in order to retrieve corresponding values for the output variables. The binding pattern is provided as a part of the signature in S_{A} and specified using the signature $R/i/j$, where R is a predicate symbol, i the arity of the predicate symbol, and j the number of input variables. We assume input variables always appear before output variables.

Example 2.2 (Physical Schema) We define a physical schema for the logical schema in Example 1. In general, we observe the following properties regarding physical implementation of DBMSs:

1. Programs/subroutines can *access* and *retrieve* data records by traversing the data structures in physical storage; and
2. Each data record is associated with a unique address in the physical storage.

For the sake of illustration, let us use a linked list data structure to store two types of records, `emp` and `dept` corresponding to `emp`loyee and `dep`artment:

```
record emp of                    record dept of
       integer    num                    integer    num
       string     name                   string     name
       reference  dept                   reference  mgr
```

Our formalism implements the above definitions of employee and department records using a set of physical predicate symbols and their access paths for $\mathsf{S}_{\mathsf{Phys}}$:

- `empfile`/1/0: set of *addresses* of `emp` records; this access path abstracts navigating a linked list (of `emp` records) in physical storage.
- `emp-num`/2/1: a set of addresses of `emp` records paired with the `emp` numbers; this access path corresponds to extracting a field (`num` in this case) from an `emp` record (given an address of such a record). The access paths `emp-name`/2/1 and `emp-dept`/2/1 and `dept-num`/2/1, `dept-name`/2/1, and `dept-mgr`/2/1 similarly abstract the field extraction of the remaining fields from the `emp` and `dept` records.

Recall that the first and second integers specify the arity and number of input variables in the binding pattern respectively, where all input variables appear before output variables in a list of arguments. We illustrate the effect of binding patterns with `emp-num`: the first argument of `emp-num` is an input argument which consumes an `emp` record address and retrieves the corresponding `num` value as output of the second argument. In addition to physical symbols corresponding directly to access paths, we can define auxiliary symbols `emp`/1 and `dept`/1 of arity 1 to represent sets of addresses of `emp` and `dept` records.

Finally, we define constraints in Σ_{Phys} to specify the connections between the logical and physical schemata:

// design of ***`emp`*** *and* ***`dept`*** *structs;* ***`emp`****/****`dept`*** *addresses, fields functional*

$$
\begin{aligned}
&\texttt{emp}(e) \rightarrow \exists y.\texttt{emp-num}(e, y),\\
&\texttt{emp}(e) \rightarrow \exists y.\texttt{emp-dept}(e, y),\\
&\texttt{emp-num}(e, y) \land \texttt{emp-num}(e, z) \rightarrow y = z,\\
&\texttt{emp-dept}(e, y) \land \texttt{emp-dept}(e, z) \rightarrow y = z,\\
&\texttt{emp-num}(y, x) \land \texttt{emp-num}(z, x) \rightarrow y = z,\\
&\texttt{emp-dept}(e, d) \rightarrow \texttt{dept}(d),\\
&\texttt{emp}(e) \rightarrow \exists y.\texttt{emp-name}(e, y),\\
&\texttt{dept}(d) \rightarrow \exists y.\texttt{dept-name}(d, y),\\
&\texttt{emp-name}(e, y) \land \texttt{emp-name}(e, z) \rightarrow y = z,\\
&\texttt{dept-name}(d, y) \land \texttt{dept-name}(d, z) \rightarrow y = z,\\
&\texttt{dept}(d) \rightarrow \exists y.\texttt{dept-num}(d, y),\\
&\texttt{dept}(d) \rightarrow \exists y.\texttt{dept-mgr}(d, y),\\
&\texttt{dept-num}(d, y) \land \texttt{dept-num}(d, z) \rightarrow y = z,\\
&\texttt{dept-mgr}(d, y) \land \texttt{dept-mgr}(d, z) \rightarrow y = z,\\
&\texttt{dept-num}(y, x) \land \texttt{dept-num}(z, x) \rightarrow y = z,\\
&\texttt{dept-mgr}(d, e) \rightarrow \texttt{emp}(e),
\end{aligned}
$$

// linked list addresses for ***emp****'s and record attributes*

$$\texttt{empfile}(x) \leftrightarrow \texttt{emp}(x),$$

// user predicates and mappings

$\texttt{employee}(x, y, z) \leftrightarrow \exists e.\texttt{baseemployee}(e, x, y, z)$,
$\texttt{emp}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \exists d.\texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z))$
$\leftrightarrow \texttt{baseemployee}(e, x, y, z)$,
$\texttt{department}(x, y, z) \leftrightarrow \exists d.\texttt{basedepartment}(d, x, y, z)$,
$\texttt{dept}(d) \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y) \wedge \exists e.(\texttt{dept-mgr}(d, e) \wedge \texttt{emp-num}(e, z))$
$\leftrightarrow \texttt{basedepartment}(d, x, y, z)$.

This completes our definition of the physical schema for Example 1.

2.1.3 Query Plans

Recall that the ultimate goal of our system is to output a program that retrieves data from physical storage for an input user query.

Definition 2.3 (Range-Restricted Formulae [2]) *Let ψ be a formula, and let R range over a set of predicate names. Let $\mathsf{Fv}(\psi)$ denote the set of all free variables in ψ. Given a set of variable names $\{x_1, x_2, \dots\}$, range-restricted formulae are defined by the following grammar:*

$$\begin{aligned}\psi ::= & R(x_1, \dots, x_k) \\ & \mid \psi_1 \wedge \psi_2 \\ & \mid \exists x_i.\psi_1 \\ & \mid \psi_1 \vee \psi_2 \quad \textit{where } \mathsf{Fv}(\psi_1) = \mathsf{Fv}(\psi_2) \\ & \mid \psi_1 \wedge \neg\psi_2 \quad \textit{where } \mathsf{Fv}(\psi_1) \supseteq \mathsf{Fv}(\psi_2)\end{aligned}$$

Definition 2.4 (Range-Restricted Schema Constraints [2]) *Let ψ and φ be range-restricted formulae, and let $x_1, \dots . x_n$ be variables. A range-restricted schema constraint is a range-restricted formula of the form $\forall x_1, \dots . x_n.(\psi \rightarrow \varphi)$, where $\mathsf{Fv}(\psi \rightarrow \varphi) = \{x_1, \dots . x_n\}$, $\mathsf{Fv}(\varphi) \subseteq \mathsf{Fv}(\psi)$.*

Definition 2.5 (Range-Restricted Query [2]) *A range-restricted query has the form* $\{(x_1, \ldots, x_n) \mid \psi\}$*, where* $\{x_1, \ldots, x_n\} = \mathsf{Fv}(\psi)$ *and* ψ *is a range-restricted formula. For the sake of convenience, we use* ψ *and* $\{(x_1, \ldots, x_n) \mid \psi\}$ *interchangeably, assuming* $(x_1, \ldots, x_n)$ *is a lexicographical ordering of the free variables.*

Definition 2.6 (Valid Query Plan [2]) *A valid query plan for user query* φ *and constraints* Σ *is a range-restricted query* ϕ *over* $\mathsf{S_A}$ *such that all binding patterns are satisfied and* $\phi \equiv \varphi$*.*

Up to this point, we have expressed everything in the DBMS in terms of first-order formulas. However, a machine cannot simply execute a query plan expressed in first-order logic. The following mapping suggests a non-trivial but straightforward procedure to transform a first-order query plan into an executable program in any language of choice:

atomic formula	$\mapsto$	access path (a `get-first` / `get-next` iterator)
conjunction	$\mapsto$	nested loops join
existential quantifier	$\mapsto$	projection (with optional duplicate information)
disjunction	$\mapsto$	concatenation
negation	$\mapsto$	simple complement

We shall revisit the issue of program generation in more details in Section 5, but first, let us illustrate the utility of our system with examples.

Example 2.7 Considering query plans and their resulting programs for the following user queries:

Q1: List employee numbers, names, and departments $(\texttt{employee}(x, y, z))$.

We can show that this user query is *logically equivalent* under the integrity constraints to the query plan $\mathsf{S_A}$:

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \\ \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z)$$

Assuming our formula to program transformation mapping, this formula would correspond to the following C-like code (with trivial simplifications and inlining of the $\texttt{ea-xxx}(x, y)$ access paths to y `:=` x`->xxx`):

```
for e in empfile do
        x := e->num;  y := e->name;
        d := e->dept; z := d->num; return (x, y, z);
```

The formula above satisfies the binding patterns associated with the access paths used as it retrieves the address of an `emp` record *before* attempting to extract the values of it's fields.

Q2: List worker numbers and names $(\exists z.\texttt{worker}(x, y, z))$.

Again, the user query is logically equivalent to the query plan:

$$\exists e, d.\texttt{empfile}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \\ \wedge \texttt{emp-dept}(e, d) \wedge \neg\texttt{dept-mgr}(d, e)$$

Note that a negation, $\neg\texttt{dept-mgr}(d, e)$, is required, and that there is no negation in the query nor in the schema that provides any direct clue that it is needed. (We are not aware of any system that can synthesize this plan given the physical data independence framework.)

Q3: List all department numbers and their names $(\exists z.\texttt{department}(x, y, z))$.

Finding a plan for this query is more difficult since we do not have a direct way to "scan" `dept` records. However, it is an easy exercise to verify that the following two formulae over $\mathsf{S_A}$ are logically equivalent to the query:

$$\exists d, e.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \\ \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y)$$

(relying on the constraint that "departments have at least one employee")

$$\exists d, e.\texttt{empfile}(e) \wedge \texttt{emp-dept}(e, d) \\ \wedge \texttt{dept-num}(d, x) \wedge \texttt{dept-name}(d, y) \wedge \texttt{dept-mgr}(d, e)$$

(relying on the constraint that "managers work in their own departments")

Both correspond to plans. However, while the second might seem to be less efficient than the first, a query optimizer should prefer it on the grounds that, in this case, the quantified variables d and e are *functionally determined* by the answer variable x. Hence, the final projection generated for the second does not need to *eliminate duplicate answers*, assuming the aforementioned translation procedure from first-order logic to executable program. This is not the case for the first of these formulae since it would return a copy of the department information for *every* employee of the department should duplicate elimination in the final projection not be performed. A more detailed discussion of duplicate elimination can be found in [33].

Many other problems and issues in physical design and query plans can be revolved in this framework, including standard RDBMS physical designs (and more), access to search structures (index access and selection), horizontal partitioning/sharding, column store/index-only plans, hash-based access to data (including hash-joins), multi-level storage (aka disk/remote/distributed files), materialized views, etc., all without any need for coding in C beyond the need for the generic specifications of `get-first` / `get-next` templates for concrete data structures.

2.2 Query Rewriting via Interpolation

2.2.1 Definability

In the previous section, we introduced the data independence property which abstracts the physical implementation of database applications from the users. Consequently, it is sensible to presume a single interpretation (i.e., database instance) of the symbols in $\mathsf{S}_{\mathsf{Log}}$, as users are oblivious to the physical implementation $\Sigma_{\mathsf{Phys}} \cup \mathsf{S}_{\mathsf{Phys}}$ of $\mathsf{S}_{\mathsf{Log}}$. Unfortunately, this seemingly innocuous presumption does not always hold. For example, *Q3* in Section 2.1.3 may have multiple valid interpretations under a fixed instance of S_{A} when the constraint "departments have at least one employee" is removed from the schema. Queries with no 'sensible' answers have undefined behaviour from the users' perspective and fall outside the scope of our discussion. To formalize the notion of 'sensible queries', we appeal to definability. To simplify notation, we hereon use finite sets of formulae X, Y in the context of $X \to Y$ to denote the conjunctions of formulae in X, Y respectively.

Proposition 2.8 (Projective Beth Definability [10]) *Let $\Sigma = \Sigma_{\mathsf{Log}} \cup \Sigma_{\mathsf{Phys}}$ be a finite set of constraints. Let $M_{|S}$ denote the reduct of model M to set S. Then the following formulations of definability condition are equivalent:*

1. *(Implicit Definability) For any two models M_1, M_2 of Σ where $M_{1|\mathsf{S}_{\mathsf{A}}} = M_{2|\mathsf{S}_{\mathsf{A}}}$, it holds that $M_1 \models q(\bar{a})$ if and only if $M_2 \models q(\bar{a})$ for all tuples $\bar{a}$ of M_1.*
2. *(Explicit Definability) q is logically equivalent to a formula ψ over S_{A}.*

Projective Beth definability also applies to an infinite set of constraints Σ. In the context of query optimization, (1) formalizes the notion of sensible queries with respect to the users' expectation: queries that are definable must have the *same answer* in *every model*

of Σ for a fixed interpretation of $\mathsf{S_A}$. Hence, projective Beth definability condition holds if and only if the user query is sensible.

In Chapter 3 and Sections 4.1 and 4.2, we show how to construct free-variable and conditional tableau proofs to verify that projective Beth definability condition holds. In order to do so, it is constructive to reformulate the definability condition as a single entailment. Let $\bullet^{\mathsf{L}}(\bullet^{\mathsf{R}})$ superscripts denote uniform renaming of atoms by adding $\mathsf{L}(\mathsf{R})$ superscripts to their predicate names respectively. Let $\Sigma_{\mathsf{A}} = \{\forall \bar{x}.(P^{\mathsf{L}}(\bar{x}) \leftrightarrow P(\bar{x}) \leftrightarrow P^{\mathsf{R}}(\bar{x})) \mid P \in \mathsf{S_A}\}$ be a finite set of constraints. Given Proposition 2.8, it is not difficult to reformulate the definability condition as the following:

$$\Sigma^{\mathsf{L}} \cup \Sigma^{\mathsf{R}} \cup \Sigma_{\mathsf{A}} \models \forall \bar{x}.(q^{\mathsf{L}}(\bar{x}) \rightarrow q^{\mathsf{R}}(\bar{x})).$$

2.2.2 Interpolation

Returning to the query optimization problem, since users expect sensible query answering behaviour, we hereon only consider definable queries. Then the following result reduces query rewriting to interpolation.

Proposition 2.9 (Craig Interpolation and Interpolant [14]) *Suppose there exists a proof of projective Beth definability condition in Proposition 2.8. Then one can construct a first order formula ψ over $\mathsf{S_A}$ such that*

$$\Sigma^{\mathsf{L}} \cup \Sigma^{\mathsf{R}} \cup \Sigma_{\mathsf{A}} \models \forall \bar{y}.(q^{\mathsf{L}}(\bar{y}) \rightarrow \psi \rightarrow q^{\mathsf{R}}(\bar{y})),$$

or equivalently, given finite sets of formulae, Σ^{L} and Σ_{A}, and $\bar{0}$ a set of Skolem constants,

$$(\Sigma^{\mathsf{L}} \cup \Sigma_{\mathsf{A}} \cup \{q^{\mathsf{L}}(\bar{0})\}) \rightarrow \psi[\bar{y}/\bar{0}] \rightarrow (\Sigma^{\mathsf{R}} \cup \Sigma_{\mathsf{A}} \rightarrow q^{\mathsf{R}}(\bar{0})).$$

We call ψ an interpolant *for q with respect to Σ and Σ_{A}.*

The second formulation in Proposition 2.9 also applies to infinite sets of formulae in the sense that if ψ is an interpolant for a finite subset of an infinite set of formulae, then it is also an interpolant for the infinite set by the compactness of first-order logic. Notice that by Definition 2.6, ψ is a valid query plan for the user query $q(\bar{x})$ if all of its binding patterns are satisfied. Moreover, Proposition 2.9 not only guarantees the existence, but also the construction of an interpolant. Hence, if a proof of definability exists, we can find a query plan for the user query $q(\bar{x})$ by constructing an interpolant such that all of its binding patterns are satisfied.

Chapter 3

Free-Variable Tableau

To construct interpolants for the purpose of query optimization, we start by considering one of the standard interpolation methods based on a family of proof systems called the analytic tableaux [19]. Tableaux-based interpolation methods usually consist of 2 phases:

1. Constructing a closed proof of the definability condition in Proposition 2.8; and
2. Applying interpolation rules to construct interpolants inductively on the structure of the proof tree.

In particular, we consider the free-variable tableau, which is a variant of the general analytic tableau. We denote the free-variable tableau system $\mathfrak{F}$. The soundness and completeness of $\mathfrak{F}$-tableau are established in [21]. To our knowledge, existing work has not applied $\mathfrak{F}$-tableau to interpolation (i.e., the construction of interpolants). In this section, we present inference and interpolation rules for $\mathfrak{F}$-tableau as well as the proof of correctness of interpolation.

3.1 Inference and Interpolation Rules

A *tableau* is a proof tree whose nodes are labelled with formulae. The *root* of a tableau represents a set of input formulae, called *premises*. A *branch* represents the conjunction of formulae appearing on it, and a *tree* represents the disjunction of its branches. A proof tree $\mathcal{F}$ is *attached at the tip* of a branch of another proof tree by appending $\mathcal{F}$ below the leaf node of that branch. Similar to the biased tableau [19], to facilitate interpolant

construction, $\mathfrak{F}$-tableau uses a signed system, which assigns each formula in the proof tree with either a *left* or *right bias*, denoted by $\mathsf{L}()$ and $\mathsf{R}()$ respectively. For the sake of brevity, whenever a rule applies to both left and right biased formulae, we denote its bias by X to represent either of the cases, and we denote its opposite bias by $\overline{\mathsf{X}}$.

Definition 3.1 ($\mathfrak{F}$-Tableau Inference Rules [19]) *Let S be a set containing all biased formulae along a branch in a $\mathfrak{F}$-tableau. Let v be a fresh variable, α, β be formulae, x a free variable in α, and $x_1, \ldots, x_n$ free variables in $\exists x.\alpha$. Let f^{X} be a fresh n-place Skolem function symbol for $\mathsf{X}(\exists x.\alpha)$ with bias X. $\mathfrak{F}$-tableau inference rules are defined as follows:*

Conjunctive

$$\frac{\{\mathsf{X}(\alpha), \mathsf{X}(\beta)\} \cup S}{\mathsf{X}(\alpha \wedge \beta) \in S}\ (\text{IC1}) \qquad \frac{\{\mathsf{X}(\neg\alpha), \mathsf{X}(\neg\beta)\} \cup S}{\mathsf{X}(\neg(\alpha \vee \beta)) \in S}\ (\text{IC2})$$

Disjunctive

$$\frac{\{\mathsf{X}(\alpha)\} \cup S \quad \{\mathsf{X}(\beta)\} \cup S}{\mathsf{X}(\alpha \vee \beta) \in S}\ (\text{ID1}) \qquad \frac{\{\mathsf{X}(\neg\alpha)\} \cup S \quad \{\mathsf{X}(\neg\beta)\} \cup S}{\mathsf{X}(\neg(\alpha \wedge \beta)) \in S}\ (\text{ID2})$$

Existential

$$\frac{\{\mathsf{X}(\alpha[x/f^{\mathsf{X}}(x_1, \ldots, x_n)])\} \cup S}{\mathsf{X}(\exists x.\alpha) \in S}\ (\text{IE1}) \qquad \frac{\{\mathsf{X}(\neg\alpha[x/f^{\mathsf{X}}(x_1, \ldots, x_n)])\} \cup S}{\mathsf{X}(\neg\forall x.\alpha) \in S}\ (\text{IE2})$$

Universal

$$\frac{\{\mathsf{X}(\alpha[x/v])\} \cup S}{\mathsf{X}(\forall x.\alpha) \in S}\ (\text{IU1}) \qquad \frac{\{\mathsf{X}(\neg\alpha[x/v])\} \cup S}{\mathsf{X}(\neg\exists x.\alpha) \in S}\ (\text{IU2})$$

We perform Skolemization on biased formulae to ensure that the original Skolem function symbols present in the premises are not biased.

Definition 3.2 (Skolemization [19]) *Let α be a formula with free variables $x, y_1, \ldots, y_n$ and f^{X} a fresh n-place function symbol with bias X. The Skolemization of $\mathsf{X}(\exists x.\alpha)$ is $\mathsf{X}(\alpha[x/f^{\mathsf{X}}(y_1, \ldots, y_n)])$. To Skolemize all $\exists$ quantifers in a formula, repeatedly perform Skolemization until the formula is free of $\exists$ quantifiers; we say that the resulting formula is* fully Skolemized.

Definition 3.3 (Maximal Skolem Term) *We call a biased Skolem term* t^{X} maximal *with respect to a set of formulae* S *if there is at least one occurrence of* t^{X} *in* S *such that* t^{X} *is not a proper subterm of another Skolem term.*

Definition 3.4 (Partial Order on Skolem Terms) *Let* t_{i_1}, t_{i_2} *be biased Skolem terms. We define a partial order* $<$ *on biased Skolem terms:* $t_{i_1} < t_{i_2}$ *if and only if* t_{i_1} *is a proper subterm of* t_{i_2}*. Given a list of biased Skolem terms* $(t_1, \ldots, t_n)$*, we say that the order of* t_{i_1} *and* t_{i_2} *satisfies the* subterm property *if* $t_{i_1} = t_j$ *and* $t_{i_2} = t_k$ *such that* $j < k$*.*

Most standard texts recommend performing Skolemization from the outside in to reduce the complexity of Skolem terms. However, the order of Skolemization does not affect equisatisfiability. The next proposition and corollary allow us to freely Skolemize input formulae before proof construction.

Proposition 3.5 (Equisatisfiability [19]) *Let* α *be a formula and* α' *the Skolemization of* α*. Then* α *is satisfiable if and only if* α' *is satisfiable.*

Corollary 3.6 Let S be a set of formulae, and let S' be the set obtained by Skolemizing each formula in S. Then there is a $\mathfrak{F}$-tableau proof that shows S is inconsistent if and only if there is a $\mathfrak{F}$-tableau proof that shows S' is inconsistent.

proof: Follows Proposition 3.5 and the soundness and completeness of $\mathfrak{F}$-tableau. ■

The defining characteristic of $\mathfrak{F}$-tableau is that it permits the introduction of free variables in universal expansions. Notably, the introduction of free variables allows universal and existential expansions to be applied in arbitrary order and without immediate ground instantiations of universally quantified variables; atomic formulae are then unified to find 'clashes' (i.e., atomic closures) for proof closure after sufficiently many inference rules have been applied. Once a substitution is applied to unify atomic formulae, inference rules may not be applied to expand the substituted proof tree, and the proof construction terminates.

Moreover, we should only consider free variables for unification and closure. To illustrate, consider substitution σ defined by $x\sigma = y$. Then, applying σ on $\forall y.(R(x, y)) \wedge P(x)$ results in $\forall y.(R(y, y)) \wedge P(y)$, but x being a free variable means that it serves as a place holder for the instantiation of another universally quantified variable, so substituting x by bound variable y is not sound.

Definition 3.7 (Free Substitution [19]) *Let* α *and* β *be formulae. A substitution* σ *being free for a formula is defined as follows:*

1. *σ is free for α whenever α is atomic;*

2. *σ is free for $\neg\alpha$ if σ is free for α;*

3. *σ is free for $(\alpha \circ \beta)$ if σ is free for α and σ is free for β, where $\circ \in \{\wedge, \vee\}$; and*

4. *Let σ_x be such that $y\sigma_x = x$ if $y = x$, and $y\sigma_x = y\sigma$ otherwise. Then σ is free for $\forall x.\alpha$ and $\exists x.\alpha$ provided that σ_x is free for α, and if y is a free variable of α other than x, $y\sigma$ does not contain x.*

We apply substitution σ to a branch b by applying σ to every formula on the branch b. We apply substitution σ to a proof tree $\mathcal{F}$ by applying σ to every branch in the proof tree $\mathcal{F}$. As a consequence of free variables and substitutions, the notions of witness and proof closure for $\mathfrak{F}$-tableau are adapted accordingly.

Definition 3.8 (Witness) *Let α be a formula and t^{X} a biased Skolem term. If t^{X} is introduced by the existential expansion of $\mathsf{X}(\exists x.\alpha)$, then we say that t^{X} is a witness for the $\exists x$ in $\mathsf{X}(\exists x.\alpha)$. If t^{X} is introduced by the Skolemization of $\exists x$ in $\mathsf{X}(\exists x.\alpha)$, then we also say that t^{X} is a witness for the $\exists x$ in $\mathsf{X}(\exists x.\alpha)$.*

Definition 3.9 (Term up to Substitution) *Let σ be a substitution, and let t denote an instance of some term in a formula ψ. We say t is a term up to substitution if we use t to denote the same instance of the term under substitution σ.*

Definition 3.10 (Scope of Witnesses) *Let t_1, t_2, t_3 be terms up to substitution. We say that t_3 is a witness inside the scope of t_1 if at any stage during $\mathfrak{F}$-tableau construction either (i) t_1 is a proper subterm of t_3, or (ii) t_3 is a witness within the scope of t_2 and t_2 a witness within the scope of t_1.*

Definition 3.11 ($\mathfrak{F}$-Tableau Closure) *Let $\mathcal{F}^v$ be a $\mathfrak{F}$-tableau, and let σ be a free substitution for $\mathcal{F}^v$ such that for every free variable x appearing in the premises (i.e., input formulae), $x\sigma = x$. A branch b in $\mathcal{F}^v$ is atomically closed under σ if $b\sigma$ contains an atom and its negation, ignoring bias labels. $\mathcal{F}$ is closed if all branches are closed under σ. We call σ a closing substitution for $\mathcal{F}^v$ and $\mathcal{F} = \mathcal{F}^v\sigma$ a closed $\mathfrak{F}$-tableau for $\mathcal{F}^v$.*

A branch is open if it is not atomically closed. A literal in $\mathcal{F}$ is open if it is not on any closed branch in $\mathcal{F}$.

Once a closed $\mathfrak{F}$-tableau is found, we can apply $\mathfrak{F}$-tableau interpolation rules to construct an interpolant inductively on the structure of the proof tree. Given a set of biased formulae

$S = \{\mathsf{L}(\alpha_0), \ldots, \mathsf{L}(\alpha_n), \mathsf{R}(\beta_0), \ldots, \mathsf{R}(\beta_k)\}$, we use $S \xrightarrow{int} \psi$ to denote $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi \to \neg\beta_0 \vee \cdots \vee \neg\beta_k$.

Definition 3.12 ($\mathfrak{F}$-Tableau Interpolation Rules) *The interpolatioin rules define the $\xrightarrow{int}$ relation inductively on the structure of the proof tree. Let $\alpha, \alpha_i, \beta, \beta_i$ denote formulae for all i. Let $\mathcal{F}$ be a closed $\mathfrak{F}$-tableau proof and $S = \{\mathsf{L}(\alpha_0), \ldots, \mathsf{L}(\alpha_n), \mathsf{R}(\beta_0), \ldots, \mathsf{R}(\beta_k)\}$ a set containing all biased formulae along a branch in $\mathcal{F}$. Let f^{X} be a fresh l-place Skolem function symbol for $\mathsf{X}(\exists x.\alpha)$ or $\mathsf{X}(\neg\forall x.\alpha)$ with bias X, $t, t_1, \ldots, t_l$ terms, v a free variable that does not appear in the premises, and r an atom.*

Base Cases

$$S \cup \{\mathsf{L}(r), \mathsf{L}(\neg r)\} \xrightarrow{int} \bot \quad \text{(B1)} \qquad S \cup \{\mathsf{R}(r), \mathsf{R}(\neg r)\} \xrightarrow{int} \top \quad \text{(B2)}$$

$$S \cup \{\mathsf{L}(r), \mathsf{R}(\neg r)\} \xrightarrow{int} r \quad \text{(B3)} \qquad S \cup \{\mathsf{R}(r), \mathsf{L}(\neg r)\} \xrightarrow{int} \neg r \quad \text{(B4)}$$

$$S \cup \{\mathsf{L}(\bot)\} \xrightarrow{int} \bot \quad \text{(B5)} \qquad S \cup \{\mathsf{R}(\bot)\} \xrightarrow{int} \top \quad \text{(B6)}$$

Conjunctive Cases

$$\frac{S \cup \{\mathsf{X}(\alpha), \mathsf{X}(\beta)\} \xrightarrow{int} \psi}{\mathsf{X}(\alpha \wedge \beta) \in S \xrightarrow{int} \psi} \text{(C1)} \qquad \frac{S \cup \{\mathsf{X}(\neg\alpha), \mathsf{X}(\neg\beta)\} \xrightarrow{int} \psi}{\mathsf{X}(\neg(\alpha \vee \beta)) \in S \xrightarrow{int} \psi} \text{(C2)}$$

Disjunctive Cases

$$\frac{S \cup \{\mathsf{L}(\alpha)\} \xrightarrow{int} \psi_1 \quad S \cup \{\mathsf{L}(\beta)\} \xrightarrow{int} \psi_2}{\mathsf{L}(\alpha \vee \beta) \in S \xrightarrow{int} \psi_1 \vee \psi_2} \text{(D1)} \qquad \frac{S \cup \{\mathsf{R}(\alpha)\} \xrightarrow{int} \psi_1 \quad S \cup \{\mathsf{R}(\beta)\} \xrightarrow{int} \psi_2}{\mathsf{R}(\alpha \vee \beta) \in S \xrightarrow{int} \psi_1 \wedge \psi_2} \text{(D2)}$$

$$\frac{S \cup \{\mathsf{L}(\neg\alpha)\} \xrightarrow{int} \psi_1 \quad S \cup \{\mathsf{L}(\neg\beta)\} \xrightarrow{int} \psi_2}{\mathsf{L}(\neg(\alpha \wedge \beta)) \in S \xrightarrow{int} \psi_1 \vee \psi_2} \text{(D3)} \qquad \frac{S \cup \{\mathsf{R}(\neg\alpha)\} \xrightarrow{int} \psi_1 \quad S \cup \{\mathsf{R}(\neg\beta)\} \xrightarrow{int} \psi_2}{\mathsf{R}(\neg(\alpha \wedge \beta)) \in S \xrightarrow{int} \psi_1 \wedge \psi_2} \text{(D4)}$$

Existential Cases

$$\frac{S \cup \{\mathsf{X}(\alpha[x/f^{\mathsf{X}}(t_1, \ldots, t_l)]\} \xrightarrow{int} \psi}{\mathsf{X}(\exists x.\alpha) \in S \xrightarrow{int} \psi} \text{(E1)} \qquad \frac{S \cup \{\mathsf{X}(\neg\alpha[x/f^{\mathsf{X}}(t_1, \ldots, t_l)]\} \xrightarrow{int} \psi}{\mathsf{X}(\neg\forall x.\alpha) \in S \xrightarrow{int} \psi} \text{(E2)}$$

Universal Cases

$$\frac{S \cup \{\mathsf{X}(\alpha[x/t]\} \xrightarrow{int} \psi}{\mathsf{X}(\forall x.\alpha) \in S \xrightarrow{int} \psi}\ (\text{U1}) \qquad \frac{S \cup \{\mathsf{X}(\neg\alpha[x/t]\} \xrightarrow{int} \psi}{\mathsf{X}(\neg\exists x.\alpha) \in S \xrightarrow{int} \psi}\ (\text{U2})$$

Quantification Cases

$$\frac{S \xrightarrow{int} \psi}{S \xrightarrow{int} \exists x.(\psi[f^{\mathsf{L}}(t_1,\ldots,t_l)/x])}\ (\text{Q1}) \qquad \frac{S \xrightarrow{int} \psi}{S \xrightarrow{int} \forall x.(\psi[f^{\mathsf{R}}(t_1,\ldots,t_l/x])}\ (\text{Q2})$$

$f^{\mathsf{L}}(t_1,\ldots,t_l)$ *does not appear in* $\beta_0,\ldots,\beta_k$ *and does not appear as a proper subterm of another Skolem term in* $\alpha_0,\ldots,\alpha_n,\psi$

$f^{\mathsf{R}}(t_1,\ldots,t_l)$ *does not appear in* $\alpha_0,\ldots,\alpha_n$ *and does not appear as a proper subterm of another Skolem term in* $\beta_0,\ldots,\beta_k,\psi$

$$\frac{S \xrightarrow{int} \psi}{S \xrightarrow{int} \exists x.(\psi[v/x])}\ (\text{Q3}) \qquad \frac{S \xrightarrow{int} \psi}{S \xrightarrow{int} \forall x.(\psi[v/x])}\ (\text{Q4})$$

v *does not appear in* S $\qquad$ v *does not appear in* S

Intuitively, a fresh free variable v may either be universally or existentially quantified since we may substitute any term, including biased Skolem terms, constants and free variables, in place of v. We illustrate this intuition behind quantification rules (Q3) and (Q4) with Example 3.13.

Example 3.13 (Fresh Free Variable Quantification) Consider the implication

$$(\forall x.R(x)) \rightarrow (\exists x.R(x)).$$

To prove the implication, we assign bias labels and start with input formulae $\mathsf{L}(\forall x.R(x))$ and $\mathsf{R}(\forall x.\neg R(x))$. Apply inference rule (IU1) to $\mathsf{L}(\forall x.R(x))$ and $\mathsf{R}(\forall x.\neg R(x))$, we get a $\mathfrak{F}$-tableau $\mathcal{F}^v$ consisting of $\{\mathsf{L}(\forall x.R(x)), \mathsf{R}(\forall x.\neg R(x))\}$ at the root and $\mathsf{L}(R(v_1)), \mathsf{R}(\neg R(v_2))$ on a single branch, where v_1, v_2 are fresh free variables.

By inspection, $\forall x.R(x)$ and $\exists x.R(x)$ are both interpolants for the implication. $\mathcal{F}$ can be closed by the closing substitution $v_1\sigma = v_2, v_2\sigma = v_2$, so we obtain a closed tableau

$\mathcal{F} = \mathcal{F}^v\sigma$. Notice that v_2 in $\mathcal{F}$ can be quantified by applying either (Q3) or (Q4), resulting in $\forall x.R(x)$ and $\exists x.R(x)$ respectively. This observation generalizes to more complicated inputs and interpolants.

Moreover, (Q1), (Q2), (Q3), and (Q4) provide as much flexibility for quantification as needed: instead of quantifying terms as soon as possible, we may delay quantification as long as the biased Skolem terms and free variables are quantified using the correct type of quantifiers within appropriate scopes based on the partial order of witnesses. Example 3.14 shows that we may delay all quantifications until the end to construct an interpolant in prenex normal form.

Example 3.14 (Delayed Quantification) Consider the implication

$$(\exists x.(P(x) \wedge \exists y.R(x,y))) \rightarrow (\exists x.(P(x) \wedge \exists y.R(x,y))).$$

To prove the implication, we assign bias labels and start with input formulae $\mathsf{L}((\exists x.(P(x) \wedge \exists y.R(x,y))))$ and $\mathsf{R}(\forall x.(\neg P(x) \vee \forall y.\neg R(x,y)))$. It is easy to verify that we can delay the application of quantification rule (Q1) until the final interpolation step and construct the interpolant $(\exists y.\exists x.(P(x) \wedge R(x,y)))$ in prenex normal form.

Finally, we remark that the universal rules (U1) and (U2) and quantification rules (Q1), (Q2), (Q3), and (Q4) are the only interpolation rules that are significantly different from the ground interpolation rules for the biased tableau presented in [19], which applies quantifiers inductively based on the structure of universal expansions in the proof tree. Notice that (Q1), (Q2), (Q3), and (Q4) rely solely on the partial order (i.e., the subterm property) and bias labels of biased Skolem terms. The new quantification rules for $\mathfrak{F}$-tableau are appropriate given the following considerations (details and proofs in Section 3.2):

1. The order of witnesses may not follow the structure of universal expansions in the proof tree because free variables and closing substitution allow universal and existential expansions to be applied in arbitrary order;
2. The order of witnesses follows the partial order of biased Skolem terms (i.e., the subterm property);
3. The type of quantifier ($\exists/\forall$) for a biased Skolem term is given by the Skolem term's bias label; and
4. Free variables that do not appear in the premises are can be either universally or existentially quantified.

3.2 Correctness of Interpolation Rules

To prove the correctness of $\mathfrak{F}$-tableau interpolation rules, we start by proving Lemmas 3.15 and 3.16, which help establish the correctness of the quantification of interpolants. We refer to free variables that that do not appear in the premises as *fresh free variables*. For an implication of the form $\phi \rightarrow \psi$, the say ϕ and ψ are the *left* and *right hand side* of the implication respectively.

Lemma 3.15 Let $\mathcal{F}^v$ be a $\mathfrak{F}$-tableau, σ a closing substitution for $\mathcal{F}^v$, and $\mathcal{F} = \mathcal{F}^v\sigma$ a closed tableau. Let $f^{\mathsf{X}}(x_1, ..., x_n)$ be a biased Skolem term in $\mathcal{F}^v$ and $t^{\mathsf{X}} = f^{\mathsf{X}}(x_1, ..., x_n)\sigma$ a biased Skolem term in $\mathcal{F}$. Then $f^{\mathsf{X}}(x_1, ..., x_n)$ does not appear in any $\overline{\mathsf{X}}$ biased formulae in $\mathcal{F}^v$. Moreover, t^{X} can only appear in a $\overline{\mathsf{X}}$ biased formula in $\mathcal{F}$ as a result of variable substitution σ for some $\overline{\mathsf{X}}$ biased universal expansion.

proof: By Definition 3.1, each biased Skolem function symbol is introduced by either Skolemization or existential expansion. By Definition 3.2, $\exists$ quantifiers in X biased formulae are always Skolemized using X biased Skolem function symbols, which are disjoint with the set of $\overline{\mathsf{X}}$ biased Skolem function symbols. By Definition 3.1, X biased existential expansions always introduce X biased Skolem function symbols, which are also disjoint with the set of $\overline{\mathsf{X}}$ biased Skolem function symbols. Hence, since f^{X} is X biased by assumption, it cannot appear in any $\overline{\mathsf{X}}$ biased formulae, or else it contradicts either Definition 3.2 or Definition 3.1. It immediately follows that t^{X} can only appear in a $\overline{\mathsf{X}}$ biased formula via substitution of free variables since inferences may not be applied after unification by Definition 3.11. ■

Lemma 3.16 Let $\mathcal{F}^v$ be a $\mathfrak{F}$-tableau, σ a closing substitution for $\mathcal{F}^v$, and $\mathcal{F} = \mathcal{F}^v\sigma$ a closed tableau. If t_i, t_j are biased Skolem terms in $\mathcal{F}$ such that t_j is a witness within the scope of t_i, then $t_i < t_j$. If t_j is a witness within the scope of fresh free variable v, then v is a proper subterm of t_j.

proof: We can find biased Skolem terms t_i^v, t_j^v in $\mathcal{F}^v$ such that $t_i = t_i^v\sigma$ and $t_j = t_j^v\sigma$. By Definition 3.11, the closing substitution σ is only applied once immediately before proof construction terminates. Then by Definition 3.10, t_j is a witness within the scope of t_i if either (1) $t_i^v < t_j^v$, or (2) $v\sigma = t_k$, where v is a free variable in t_j^v and $t_i \leq t_k$. Since $\mathfrak{F}$-tableau never interleaves inferences with substitutions, biased Skolem term creation in $\mathcal{F}^v$ is acyclic, so in case (1), $t_i^v < t_j^v$ implies that $t_i < t_j$. In case (2), since σ is applied to v only once in the end, we get $t_i < t_j$. The proof for v being a proper subterm of t_j is analogous. ■

Now we are ready to prove the correctness of $\mathfrak{F}$-tableau interpolation rules. The main difficulty in proving interpolation rules for $\mathfrak{F}$-tableau is that the order of witnesses, hence quantification, does not always follow the order of existential and universal expansions in the proof tree. Hence, unlike the biased tableau [19], we cannot simply quantify the interpolant based on the structure of the proof tree but have to rely on the order of witnesses (i.e., the subterm property) to quantify biased Skolem terms and fresh free variables in the correct order.

Lemma 3.17 Let $\mathcal{F}$ be a closed $\mathfrak{F}$-tableau and $S = \{\mathsf{L}(\alpha_0), \ldots, \mathsf{L}(\alpha_n), \mathsf{R}(\beta_0), \ldots, \mathsf{R}(\beta_k)\}$ a set containing all biased formulae along a branch in $\mathcal{F}$. If $S \xrightarrow{int} \psi$ by $\mathfrak{F}$-tableau interpolation rules from Definition 3.1, then the following invariants hold:

1. $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi$ and $\psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$; and
2. Constant, function and relational symbols in ψ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ or appear in a biased Skolem term.

proof: We prove that the above invariants hold for $\mathfrak{F}$-tableau interpolation rules.

Base cases: For (B1), we know that $r \wedge \neg r \rightarrow \bot$, and $\bot$ implies an arbitrary formula, so $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge r \wedge \neg r \rightarrow \bot \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. For (B2), we know that arbitrary formula implies $\top$, and $\top \rightarrow \neg r \vee r$, so $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \top \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee r \vee \neg r$. For (B3) and (B4), since $r \rightarrow r$ and $\neg r \rightarrow \neg r$ are tautologies, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge r \rightarrow r \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee r$ and $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \neg r \rightarrow \neg r \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg r$ respectively. Finally, for (B5) and (B6), we know that $\bot \rightarrow \bot$ and $\top \rightarrow \top$ are tautologies, $\bot$ implies an arbitrary formula, and arbitrary formula implies $\top$, so we get $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \bot \rightarrow \bot \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$ and $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \top \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \top$ respectively. Hence, invariant (1) holds for all base cases. Invariant (2) holds for all base cases since $\top$ and $\bot$ are basic logical symbols, and (B3) and (B4) only apply when r appears both L and R biased.

Left conjunctive cases: Let us prove the conjunctive interpolation rule (C1) for $\mathsf{X} = \mathsf{L}$. Suppose invariant properties hold for $S \cup \{\mathsf{L}(\alpha), \mathsf{L}(\beta)\} \xrightarrow{int} \psi$. We prove that the invariant properties also hold for $S \xrightarrow{int} \psi$, where $\mathsf{L}(\alpha \wedge \beta) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha \wedge \beta \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so we get $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge (\alpha \wedge \beta) \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Hence, since $\mathsf{L}(\alpha \wedge \beta) \in S$, invariant (1) holds for $S \xrightarrow{int} \psi$. By invariant (2), all constant, function and relational symbols in ψ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha \wedge \beta$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ or in a biased Skolem term. Since $\alpha \wedge \beta \in S$, invariant (2) holds for $S \xrightarrow{int} \psi$. The correctness of (C2) for $\mathsf{X} = \mathsf{L}$ follows by replacing $\alpha, \beta, \alpha \wedge \beta$ by $\neg\alpha, \neg\beta, \neg(\alpha \vee \beta)$. Thus, both invariants hold for all left conjunctive cases.

Right conjunctive cases: Let us prove the conjunctive interpolation rule (C1) for $\mathsf{X} = \mathsf{R}$. Suppose invariant properties hold for $S \cup \{\mathsf{R}(\alpha), \mathsf{R}(\beta)\} \xrightarrow{int} \psi$. We prove that the invariant properties also hold for $S \xrightarrow{int} \psi$, where $\mathsf{R}(\alpha \wedge \beta) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\alpha \vee \neg\beta$, so we get $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg(\alpha \wedge \beta)$. Hence, since $\mathsf{R}(\alpha \wedge \beta) \in S$, invariant (1) holds for $S \xrightarrow{int} \psi$. By invariant (2), all constant, function and relational symbols in ψ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\alpha \vee \neg\beta$ or in a biased Skolem term. Since $\mathsf{R}(\alpha \wedge \beta) \in S$, invariant (2) holds for $S \xrightarrow{int} \psi$. The correctness of (C2) for $\mathsf{X} = \mathsf{R}$ follows by replacing $\alpha, \beta, \alpha \wedge \beta$ by $\neg\alpha, \neg\beta, \neg(\alpha \vee \beta)$. Thus, both invariants hold for all right conjunctive cases.

Left disjunctive cases: Let us prove the left disjunctive interpolation rule (D1). Suppose invariants hold for $S \cup \{\mathsf{L}(\alpha)\} \xrightarrow{int} \psi_1$ and $S \cup \{\mathsf{L}(\beta)\} \xrightarrow{int} \psi_2$. We prove that invariants hold for $S \xrightarrow{int} \psi_1 \vee \psi_2$, where $\mathsf{L}(\alpha \vee \beta) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha \rightarrow \psi_1 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$ and $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \beta \rightarrow \psi_2 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Then we get $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge (\alpha \vee \beta) \rightarrow \psi_1 \vee \psi_2 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Simplify, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi_1 \vee \psi_2 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so invariant (1) holds for $S \xrightarrow{int} \psi_1 \vee \psi_2$. By invariant (2), all relational, function and constant symbols in $\psi_1(\psi_2)$ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha(\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \beta)$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ respectively or in a biased Skolem term. Then all relational, function and constant symbols in $\psi_1 \vee \psi_2$ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ or in a biased Skolem term, so invariant (2) holds for $S \xrightarrow{int} \psi_1 \vee \psi_2$. The correctness of the left conjunctive interpolation rule (D3) follows by replacing $\alpha, \beta, \alpha \vee \beta$ by $\neg\alpha, \neg\beta, \neg(\alpha \wedge \beta)$. Thus, both invariants hold for all left disjunctive cases.

Right disjunctive cases: Let us prove the right disjunctive interpolation rule (D2). Suppose invariants hold for $S \cup \{\mathsf{R}(\alpha)\} \xrightarrow{int} \psi_1$ and $S \cup \{\mathsf{R}(\beta)\} \xrightarrow{int} \psi_2$. We prove that invariants hold for $S \xrightarrow{int} \psi_1 \wedge \psi_2$, where $\alpha \vee \beta \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi_1 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\alpha$ and $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi_2 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\beta$. Then since $\neg\alpha \wedge \neg\beta \equiv \neg(\alpha \vee \beta)$, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi_1 \wedge \psi_2 \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg(\alpha \vee \beta)$, so invariant (1) holds for $S \xrightarrow{int} \psi_1 \wedge \psi_2$. Analogous to the proof of the left disjunctive cases, invariant (2) holds for $S \xrightarrow{int} \psi_1 \wedge \psi_2$. The correctness of the right conjunctive interpolation rule (D4) follows by replacing $\alpha, \beta, \alpha \vee \beta$ by $\neg\alpha, \neg\beta, \neg(\alpha \wedge \beta)$. Thus, both invariants hold for all right disjunctive cases.

Left existential cases: Let us prove the existential rule (E1) for $\mathsf{X} = \mathsf{L}$. Suppose invariants hold for $S \cup \{\mathsf{L}(\alpha[x/f^{\mathsf{L}}(t_1, \ldots, t_l)])\} \xrightarrow{int} \psi$. We prove that invariants hold for $S \xrightarrow{int} \psi$, where $\mathsf{L}(\exists x.\alpha) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha[x/f^{\mathsf{L}}(t_1, \ldots, t_l)] \rightarrow$

$\psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Then since $\exists x.\alpha \to \alpha[x/f^{\mathsf{L}}(t_1,\ldots,t_l)]$ and $\mathsf{L}(\exists x.\alpha) \in S$, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so invariant (1) holds. By invariant (2), all constant, function and relational symbols in ψ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha[x/f^{\mathsf{L}}(t_1,\ldots,t_l)]$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ or in a biased Skolem term. Since $f^{\mathsf{L}}(t_1,\ldots,t_l)$ only contains symbols appearing in the original premise, new Skolem function symbols and fresh free variables, invariant (2) still holds after removing $f^{\mathsf{L}}(t_1,\ldots,t_l)$ from S. The correctness of (E2) follows from replacing $\alpha[x/f^{\mathsf{L}}(t_1,\ldots,t_l)]$ and $\exists x.\alpha$ by $\neg\alpha[x/f^{\mathsf{L}}(t_1,\ldots,t_l)]$ and $\neg\forall x.\alpha$ respectively.

Right existential cases: Let us prove the existential rule (E1) for $\mathsf{X} = \mathsf{R}$. Suppose invariants hold for $S \cup \{\mathsf{R}(\alpha[x/f^{\mathsf{R}}(t_1,\ldots,t_l)])\} \xrightarrow{int} \psi$. We prove that invariants hold for $S \xrightarrow{int} \psi$, where $\mathsf{R}(\exists x.\alpha) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\alpha[x/f^{\mathsf{R}}(t_1,\ldots,t_l)]$. Then since $\neg(\exists x.\alpha) \to \neg\alpha[x/f^{\mathsf{R}}(t_1,\ldots,t_l)]$ and $\mathsf{R}(\exists x.\alpha) \in S$, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi_1 \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so invariant (1) holds. Analogous to the proof for the left universal case, invariant (2) holds. The correctness of (E2) follows from replacing $\alpha[x/f^{\mathsf{R}}(t_1,\ldots,t_l)]$ and $\exists x.\alpha$ by $\neg\alpha[x/f^{\mathsf{R}}(t_1,\ldots,t_l)]$ and $\neg\forall x.\alpha$ respectively.

Left universal cases: Let us prove the universal rule (U1) for $\mathsf{X} = \mathsf{L}$. Suppose invariants hold for $S \cup \{\mathsf{L}(\alpha[x/t])\} \xrightarrow{int} \psi$. We prove that invariants hold for $S \xrightarrow{int} \psi$, where $\mathsf{L}(\forall x.\alpha) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha[x/t] \to \psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Then since $\forall x.\alpha \to \alpha[x/t]$ and $\mathsf{L}(\forall x.\alpha) \in S$, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so invariant (1) holds. By invariant (2), all constant, function and relational symbols in ψ are either in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n \wedge \alpha[x/t]$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$ or in a biased Skolem term. Since c only contains symbols appearing in the original premise, new Skolem function symbols and fresh free variables, invariant (2) still holds after removing t from S. The correctness of (U2) follows from replacing $\alpha[x/t]$ and $\forall x.\alpha$ by $\neg\alpha[x/t]$ and $\neg\exists x.\alpha$ respectively.

Right universal cases: Let us prove the universal rule (U1) for $\mathsf{X} = \mathsf{R}$. Suppose invariants hold for $S \cup \{\mathsf{R}(\alpha[x/t])\} \xrightarrow{int} \psi$. We prove that invariants hold for $S \xrightarrow{int} \psi$, where $\mathsf{R}(\forall x.\alpha) \in S$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi \to \neg\beta_1 \vee \cdots \vee \neg\beta_k \vee \neg\alpha[x/t]$. Then since $\forall x.\alpha \to \alpha[x/t] \equiv \neg\alpha[x/t] \to \neg\forall x.\alpha$ and $\mathsf{R}(\forall x.\alpha) \in S$, we get $\alpha_0 \wedge \cdots \wedge \alpha_n \to \psi_1 \to \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so invariant (1) holds. Analogous to the proof for the left universal case, invariant (2) holds. The correctness of (U2) follows from replacing $\alpha[x/t]$ and $\forall x.\alpha$ by $\neg\alpha[x/t]$ and $\neg\exists x.\alpha$ respectively.

Left biased Skolem quantification cases: Let us prove the left biased Skolem quantification rule (Q1). Suppose invariants hold for $S \xrightarrow{int} \psi$, where L biased Skolem t does not appear in $\beta_0,\ldots,\beta_k$, and t does not appear as a proper subterm of another Skolem term

in $\alpha_0, \ldots, \alpha_t, \psi$. We prove that invariants hold for $S \xrightarrow{int} \exists x.\psi[t/x]$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Since c does not appear as a proper subterm of another Skolem term in ψ, $\alpha_0, \ldots, \alpha_t, \beta_0, \ldots, \beta_k$, by Lemma 3.16, all biased Skolem terms within the scope of t have already been quantified, so we may quantify t. Then since $\psi \rightarrow \exists x.\psi[t/x]$, we get $\alpha_0, \ldots, \alpha_t \rightarrow \exists x.\psi[t/x]$, so the left hand side of implication holds. For the right hand side of implication, since t does not appear in $\beta_0, \ldots, \beta_k$ and every appearance of t in R biased formulae is a result of variable substitution by Lemma 3.15, we have $\beta_1 \wedge \cdots \wedge \beta_k \rightarrow \forall x.(\neg\psi[t/x])$, so we get $\exists x.\psi[t/x] \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Combining both side, invariant (1) holds for $S \xrightarrow{int} \exists x.\psi[t/x]$. Since invariant (2) holds for $S \xrightarrow{int} \psi$ and t is removed from ψ, invariant (2) holds.

Right biased Skolem quantification cases: Let us prove the right biased Skolem quantification rule (Q2). Suppose invariants hold for $S \xrightarrow{int} \psi$, where R biased Skolem t does not appear in $\alpha_0, \ldots, \alpha_n$, and t does not appear as a proper subterm of another Skolem term in $\beta_0, \ldots, \beta_k, \psi$. We prove that invariants hold for $S \xrightarrow{int} \forall x.\psi[t/x]$. By invariant (1), we have $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. Since t does not appear as a proper subterm of another Skolem term in ψ, $\beta_0, \ldots, \beta_k, \alpha_0, \ldots, \alpha_n$, by Lemma 3.16, all biased Skolem terms within the scope of t have already been quantified, so we may quantify t. Then since $\neg\psi \rightarrow \exists x.(\neg\psi[t/x])$, we get $\forall x.\psi[t/x] \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$, so the right hand side of implication holds. For the left hand side of implication, since t does not appear in $\alpha_0, \ldots, \alpha_n$ and every appearance of t in L biased formulae is a result of variable substitution by Lemma 3.15, we have $\alpha_0 \wedge \cdots \wedge \alpha_t \rightarrow \forall x.\psi[t/x]$. Combining both sides, invariant (1) holds. Since invariant (2) holds for $S \xrightarrow{int} \psi$ and t is removed from ψ, invariant (2) holds.

Fresh free variable quantification cases: Let us prove the fresh free variable quantification rules (Q3) and (Q4). Suppose invariants hold for $S \xrightarrow{int} \psi$, where v is a fresh free variable in ψ not in S. Then since free variable v is fresh, v can be quantified using either universal or existential. Then, since v does not appear in S, we get $\alpha_0, \ldots, \alpha_n \rightarrow \forall x.\psi[v/x] \rightarrow \beta_0, \ldots, \beta_k$ in (Q3) and $\alpha_0, \ldots, \alpha_n \rightarrow \exists x.\psi[v/x] \rightarrow \beta_0, \ldots, \beta_k$ in (Q4). Since invariant (2) holds for $S \xrightarrow{int} \psi$ and v is removed from ψ, invariant (2) holds.

This covers all the cases. Therefore, invariant properties hold for all $\mathfrak{F}$-tableau interpolation rules. ■

The following lemma ensures that all biased Skolem terms and fresh free variables can be quantified in the final interpolant.

Lemma 3.18 Let S be a set of biased input formulae, and suppose S is the root of a closed tableau $\mathcal{F}$. If $S \xrightarrow{int} \psi'$ by $\mathfrak{F}$-tableau interpolation on $\mathcal{F}$, then there is a formula ψ

such that $S \xrightarrow{int} \psi$ and ψ does not contain any biased Skolem terms or fresh free variables.

proof: By quantification rules, we can always apply (Q1) and (Q2) to quantify all biased Skolem terms in ψ'. We can then apply (Q3) and (Q4) to quantify all fresh free variables in ψ' to obtain ψ. Hence, we can always construct an interpolant that is free of biased Skolem terms and fresh free variables when an interpolant exists. ■

By Lemma 3.18, we can assume from hereon that all biased Skolem terms and fresh free variables have been quantified in ψ given $S \xrightarrow{int} \psi$.

Theorem 3.19 (Correctness of $\mathfrak{F}$-Tableau Interpolation Rules) Let $S = \{\mathsf{L}(\alpha_0), \ldots, \mathsf{L}(\alpha_n), \mathsf{R}(\beta_0), \ldots, \mathsf{R}(\beta_k)\}$ be a set of formulae at the root of a $\mathfrak{F}$-tableau $\mathcal{F}^v$. Let σ be a closing substitution for $\mathcal{F}^v$, and let $\mathcal{F} = \mathcal{F}^v\sigma$ be a closed tableau. If $S \xrightarrow{int} \psi$ by $\mathfrak{F}$-tableau interpolation on $\mathcal{F}$, then ψ is an interpolant for the sentence $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$.

proof: By invariant (1) in Lemma 3.17, $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \psi \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. By Lemma 3.18 and invariant (2) in Lemma 3.17, every constant, function, variable and relational symbol in ψ is in the common language of $\alpha_0 \wedge \cdots \wedge \alpha_n$ and $\neg\beta_1 \vee \cdots \vee \neg\beta_k$. Hence, by definition, ψ is an interpolant for $\alpha_0 \wedge \cdots \wedge \alpha_n \rightarrow \neg\beta_1 \vee \cdots \vee \neg\beta_k$. ■

Finally, Lemma 3.20 generalizes the correctness of interpolation rules to Skolemized input formulae.

Lemma 3.20 (Skolemization Preserves Interpolant) Let S_1, S_2 be finite sets of formulae, S_1', S_2' their respective Skolemizations, and ψ a formula that does not contain any biased Skolem terms. Then ψ is an interpolant for $S_1 \rightarrow S_2$ if and only if ψ is an interpolant for $S_1' \rightarrow S_2'$.

proof: Since ψ does not contain any biased Skolem terms, all constant, function and relational symbols in ψ appear in the common language of S_1 and S_2. By Proposition 3.5, S_1 and S_1' are equisatisfiable, so $S_1 \rightarrow \psi$ if and only if $S_1' \rightarrow \psi$. Similarly, we have $\psi \rightarrow S_2$ if and only if $\psi \rightarrow S_2'$. Hence, $S_1' \rightarrow \psi \rightarrow S_2'$ if and only if $S_1 \rightarrow \psi \rightarrow S_2$, as desired. ■

As a consequence of Craig Interpolation (Proposition 2.9), we can construct query plans (i.e., interpolants) for definable user queries via $\mathfrak{F}$-tableau interpolation from input formulae $\{\mathsf{L}(\Sigma^{\mathsf{L}} \cup \Sigma_{\mathsf{A}} \cup \{q^{\mathsf{L}}(\overline{0})\}), \mathsf{R}(\Sigma^{\mathsf{R}} \cup \Sigma_{\mathsf{A}} \cup \{\neg q^{\mathsf{L}}(\overline{0})\}\}$, where a set of formulae is shorthand for the conjunction of formulae in the set.

Example 3.21 ($\mathfrak{F}$-Tableau Interpolation) We use $\mathfrak{F}$-tableau to construct an interpolant for $X \longrightarrow Y$. Let $X = \{R^{\mathsf{L}}(v, 0), \forall x.(R^{\mathsf{L}}(x, 0) \rightarrow \exists y, z.P(x, y, z) \wedge O(x))\}$ and $Y =$

$\{\forall x, y, z.(P(x, y, z) \wedge O(x) \rightarrow R^{\mathsf{R}}(x, 0)) \rightarrow R^{\mathsf{R}}(v, 0)\}$, where v is an initial free variable and 0 is a special Skolem constant. To initialize the $\mathfrak{F}$-tableau, we assigned bias labels to formulae and Skolemize $\exists y$ as follows:

$$S = \{\mathsf{L}(R^{\mathsf{L}}(v, 0)),\ \mathsf{L}(\forall x.(R^{\mathsf{L}}(x, 0) \rightarrow \exists z.P(x, f_y^{\mathsf{L}}(x, z), z) \wedge O(x))),$$
$$\mathsf{R}(\forall x, y, z.(P(x, y, z) \wedge O(x) \rightarrow R^{\mathsf{R}}(x, 0))),\ \mathsf{R}(\neg R^{\mathsf{R}}(v, 0))\}.$$

Then, we construct a $\mathfrak{F}$-tableau proof for S, find a closing substitution, and construct an interpolant inductively on the structure of the proof tree from its closed proof as shown in Figure 3.1.

Although it is feasible to use $\mathfrak{F}$-tableau for query optimization, this approach is problematic due to the following drawbacks:

1. Each $\mathfrak{F}$-tableau proof corresponds to a single interpolant, but the goal is to enumerate different interpolants (i.e., query plans);
2. Finding alternative interpolants requires backtracking and unification which can lead to performance issues; and
3. The interpolant constructed depends on the syntactic structures of the input formulae, so naïve backtracking will not work in principle.

To enumerate query plans efficiently, in the next chapter, we introduce another tableau variant called the condition tableau which addresses all of the above issues.

Closing Substitution

$$\sigma(x') = v,\ \ \sigma(x'') = v,\ \ \sigma(y') = f^{\mathsf{L}}_y(v, f^{\mathsf{L}}_z(v)),\ \ \sigma(z') = f^{\mathsf{L}}_z(v)$$

$\mathfrak{F}$-Tableau and Interpolation

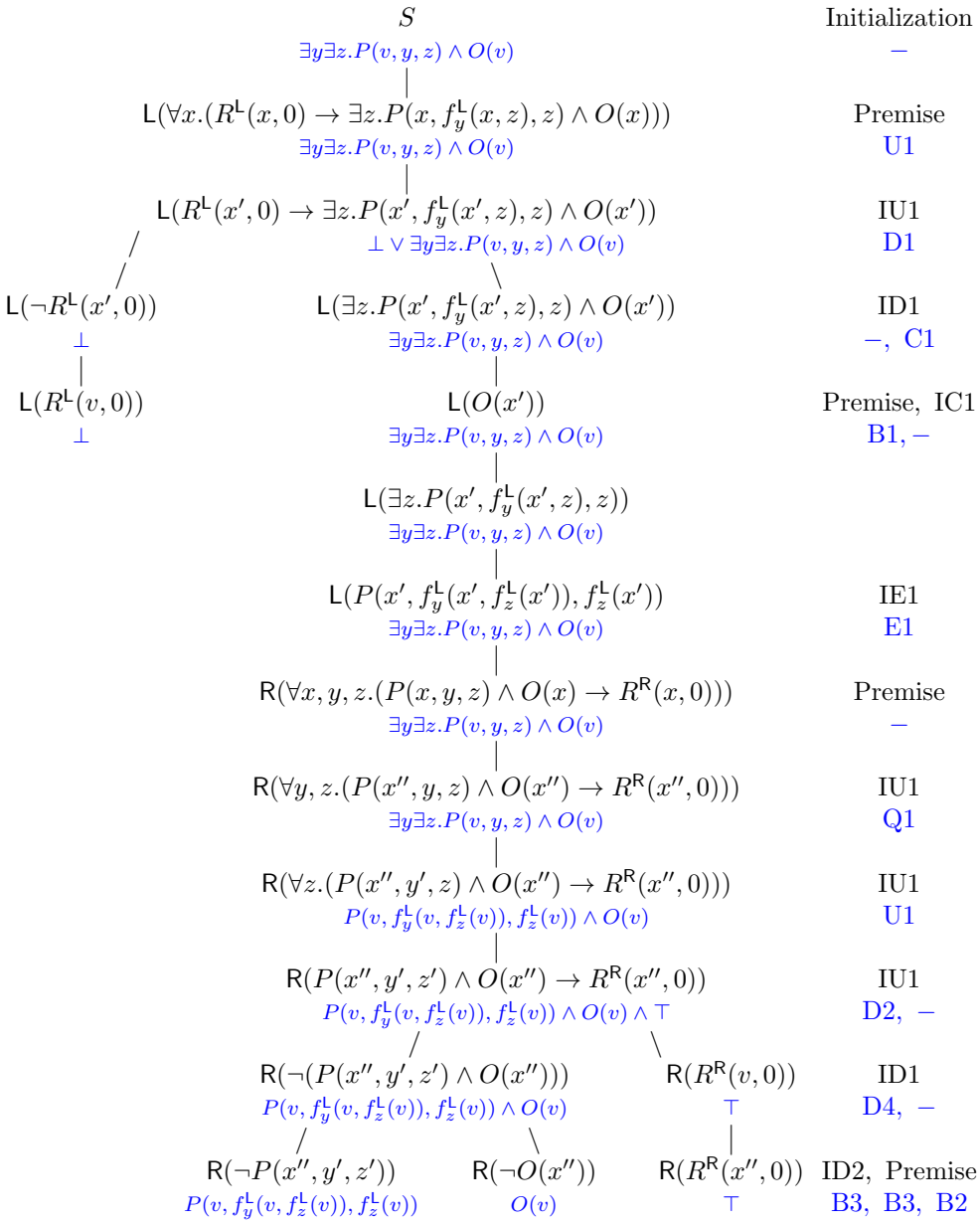


Figure 3.1: A free-variable tableau.

Chapter 4

Conditional Tableau

To address the drawbacks of $\mathfrak{F}$-tableau interpolation, we introduce the conditional tableau which improves interpolant enumeration for the purpose of query optimization. We denote the conditional tableau system $\mathfrak{K}$. The intuition behind $\mathfrak{K}$-tableau is as follows: instead of constructing a single proof tree with fixed structural properties, we remember partial information about past derivations to represent many tableau proof trees simultaneously by a single $\mathfrak{K}$-tableau. Consequently, the interpolation mechanism is adapted accordingly: inductive construction is replaced by a 'meet-in-the-middle' approach that (i) extracts common features among many tableau proof trees from a single $\mathfrak{K}$-tableau proof tree, then (ii) checks whether an externally supplied formula is an interpolant based on the features extracted from the $\mathfrak{K}$-tableau. To focus on $\mathfrak{K}$-tableau, in this chapter, we assume that a black box supplies all candidate interpolants externally. We discuss how to generate candidate interpolants in Chapter 5.

The remainder of Chapter 4 is organized as follows. First, we describe $\mathfrak{K}$-tableau by defining its input normalization, inference rules, and interpolation mechanism. Then, we prove the correctness of its inference rules and interpolation mechanism by reduction to $\mathfrak{F}$-tableau, which is defined in Chapter 3. Lastly, we discuss how to effectively apply a set of special $\mathfrak{K}$-tableau rules, called physical rules, to optimize theorem proving and enumerate interpolants that are otherwise not found.

4.1 Description

4.1.1 Schema Normalization

Normalization is a common optimization technique in automated theorem proving [26, 30]. Preprocessing input premises admits two main advantages: (1) fewer and simpler inference rules; and (2) optimization during theorem proving. For our purposes, we appeal to standard absorptions [23] and define absorption normal form (ANF) for input constraints. Roughly speaking, ANF captures range-restricted constraints of the form $\forall \bar{x}.(R_1(\bar{x}_1) \land \cdots \land R_n(\bar{x}_n) \to P_1(\bar{x}_{n+1}) \lor \cdots \lor P_k(\bar{x}_{n+k}))$, where $R_1, \ldots, R_n, P_1, \ldots, P_k$ are predicate symbols, $\bar{x}_i$ a set of variables for each i, and $\bar{x}$ the set of all free variables in the constraint. A constraint in ANF only needs to be expanded if all $R_1, \ldots, R_n$ match with positive ground instances. Moreover, instantiations of all universally quantified variables $\bar{x}$ are determined by matching positive ground instances of $R_1, \ldots, R_n$.

Definition 4.1 (Absorption Normal Form (ANF) [25]) *The set of ANF formulae, Q, is defined by the following grammar:*

$$
\begin{array}{lcl}
A & ::= & R(\bar{x}) \\
C & ::= & A \mid C \land C \\
Q & ::= & \forall \bar{x}.(C \to A) \mid \forall \bar{x}.(C \to \bot) \mid \forall \bar{x}.(A \to A \lor A) \mid \forall \bar{x}.(A \to \exists v.R(v, \bar{x}))
\end{array}
$$

where R ranges over a set of predicate symbols, Q is range-restricted without free variables.

Proposition 4.2 (Correctness of Absorption [23]) *Let S be a set of constraints in ANF. We say a tableau proof is restricted if every constraint $\forall \bar{x}.(R_1(\bar{x}_1) \land \cdots \land R_n(\bar{x}_n) \to P_1(\bar{x}_{n+1}) \lor \cdots \lor P_k(\bar{x}_{n+k}))$ is expanded only if all $R_1, \ldots, R_n$ match with positive ground instances. Then there is a tableau proof that shows S is inconsistent if and only if there is a restricted tableau proof that shows S is inconsistent.*

Given a range-restricted user query α_q and schema consisting of range-restricted constraints, we can normalize the user query and constraints at once by introducing a new constraint $q \leftrightarrow \alpha_q$, where q is a special atom reserved for formulating the user query, then initializing the tableau with the ground atom q. Moreover, we can apply the same technique to remove constraints of the form $\bot \to R()$ by adding ground atom $R()$ to the initial tableau. Then a set of range-restricted input constraints can always be transformed into a set of ANF formulae via the following normalization procedure. For the sake of convenience, let $\mathsf{Fv}(\alpha)$ denote the set of free variables in α; we omit writing out $\forall$ quantification for $\mathsf{Fv}(\psi \to \varphi)$, and assume they are universally quantified.

Procedure 4.3 (ANF Normalization) Let $\alpha, \beta, \alpha_1, \alpha_2, \beta_1, \beta_2, \alpha_1', \alpha_2', \beta', \beta''$ denote formulae. Let R_i be predicate symbols and $\bar{x}, \bar{x}_i$ variables for all i. We treat $\bot$ as an atom with empty arguments. A range-restricted constraint $\alpha \to \beta$ can be transformed into a set of logically equivalent formulae in ANF by a mutually recursive normalization procedure that begins with $\mathsf{NormRight}(\alpha \to \beta)$. The mutually recursive procedures $\mathsf{NormRight}$ and $\mathsf{NormLeft}$ are defined as follows:

Normalize right hand side:

1. $\mathsf{NormRight}(\alpha \to R_1(\bar{x}_1)) = \mathsf{NormLeft}(\alpha \to R_1(\bar{x}_1))$
2. $\mathsf{NormRight}(\alpha \to \exists v.\beta) = \mathsf{NormLeft}(\alpha \to P_1(\bar{x})) \cup \mathsf{NormLeft}(P_1(\bar{x}) \to \exists v.P_2(v, \bar{x})) \cup \mathsf{NormRight}(P_2(v, \bar{x}) \to \beta)$, where P_1, P_2 are fresh predicates for $\exists v.\beta$ and $\bar{x} = \mathsf{Fv}(\beta) \backslash \{v\}$
3. $\mathsf{NormRight}(\alpha \to \beta_1 \wedge \neg\beta_2) = \mathsf{NormRight}(\alpha \to \beta_1) \cup \mathsf{NormRight}(\alpha \wedge \beta_2 \to \bot)$
4. $\mathsf{NormRight}(\alpha \to \beta_1 \wedge \beta_2) = \mathsf{NormRight}(\alpha \to \beta_1) \cup \mathsf{NormRight}(\alpha \to \beta_2)$
5. $\mathsf{NormRight}(\alpha \to \beta_1 \vee \beta_2) = \mathsf{NormLeft}(\alpha \to P_1(\bar{x})) \cup \mathsf{NormLeft}(P_1(\bar{x}) \to P_2(\bar{x}) \vee P_3(\bar{x})) \cup \mathsf{NormRight}(P_2(\bar{x}) \to \beta_1) \cup \mathsf{NormRight}(P_3(\bar{x}) \to \beta_2)$, where P_1 is a fresh predicates for $\beta_1 \vee \beta_2$, P_2, P_3 fresh for β_1, β_2 and $\bar{x} = \mathsf{Fv}(\beta_1) \cup \mathsf{Fv}(\beta_2)$

Normalize left hand side:

1. $\mathsf{NormLeft}(R_1(\bar{x}_1) \wedge \cdots \wedge R_k(\bar{x}_k) \to R_{k+1}(\bar{x}_{k+1})) = \{R_1(\bar{x}_1) \wedge \cdots \wedge R_k(\bar{x}_k) \to R_{k+1}(\bar{x}_{k+1})\}$
2. $\mathsf{NormLeft}(R_1(\bar{x}_1) \to \exists v.R_2(\bar{x}_1)) = \{R_1(\bar{x}_1) \to \exists v.R_2(\bar{x}_1)\}$
3. $\mathsf{NormLeft}(R_1(\bar{x}_1) \to R_2(\bar{x}_2) \vee R_3(\bar{x}_2)) = \{R_1(\bar{x}_1) \to R_2(\bar{x}_2) \vee R_3(\bar{x}_2)\}$
4. $\mathsf{NormLeft}(\alpha \wedge \bot \to \beta) = \{\alpha \to \beta\}$
5. $\mathsf{NormLeft}(\alpha_1 \wedge \neg\alpha_2 \to \beta) = \mathsf{NormRight}(\alpha_1 \to \beta \vee \alpha_2)$
6. $\mathsf{NormLeft}(\alpha_1 \vee \alpha_2 \to \beta) = \mathsf{NormLeft}(\alpha_1 \to \beta) \cup \mathsf{NormLeft}(\alpha_2 \to \beta)$
7. $\mathsf{NormLeft}(\alpha_1 \wedge \alpha_2 \to \beta) = \{\alpha_1' \wedge \alpha_2' \to \beta'' \mid \alpha_1' \to \beta' \in \mathsf{NormLeft}(\alpha_1 \to \beta), \alpha_2' \to \beta'' \in \mathsf{NormLeft}(\alpha_2 \to \beta')\}$

proof of correctness: Assume the input formula is a range-restricted constraint. First, we show that NormRight recurs on a set of logically equivalent, range-restricted formulae. We can prove that the same holds for NormLeft.

Case (1): Statement holds since the input formula is unchanged, so it is logically equivalent to itself and still range-restricted.

Case (2): Since P_1, P_2 are fresh predicates and $\bar{x} = \mathsf{Fv}(\beta)\backslash\{v\} = \mathsf{Fv}(\exists v.\beta)$, we have $\alpha \to \exists v.\beta \equiv \{\alpha \to P_1(\bar{x}), P_1(\bar{x}) \to \exists v.P_2(v,\bar{x}), P_2(v,\bar{x}) \to \beta\}$. Also, since $\alpha \to \exists v.\beta$ is range-restricted if and only if $\alpha, \exists v.\beta$ are range-restricted and $\mathsf{Fv}(\exists v.\beta) \subseteq \mathsf{Fv}(\alpha)$ by Definition 2.4, we have that each constraint in $\{\alpha \to P_1(\bar{x}), P_1(\bar{x}) \to \exists v.P_2(v,\bar{x}), P_2(v,\bar{x}) \to \beta\}$ is also range-restricted. Hence, statement holds for case (2).

Cases (3), (4), (5): Proofs are analogous to the proof of case (2).

We can prove that the same holds for NormLeft. For cases (1)-(6), the proofs are analogous to the proof of NormRight case (2). For case (7), logical equivalence and range-restrictedness follow from the fact that $\mathsf{NormLeft}(\alpha_1 \to \beta)$ is equivalent to normalizing $\alpha_1 \wedge \alpha_2 \to \beta$ while keeping α_2 unchanged and $\mathsf{NormLeft}(\alpha_2 \to \beta')$ equivalent to normalizing $\alpha_1' \wedge \alpha_2 \to \beta'$ while keeping α_1' unchanged.

Second, we show that each non-mutually recursive call either terminates in a base case or in a mutually recursive call. For NormRight, the right hand sides of input formulae cover all range-restricted formulae by Definition 2.3. For NormLeft, the left hand sides of input formulae cover all range-restricted formulae by Definition 2.3, and the right hand sides of input formulae cover all input formulae from NormRight's recursive call by NormRight cases (1), (2) and (5). By inspection, every non-mutually recursive call decreases the degree of the input formula by at least 1, so each non-mutually recursive call either terminates in a mutually recursive call or in a base case.

Third, we show that each mutually recursive call terminates in a base case. Notice that the only mutually recursive call for NormLeft is case (5). By inspection, NormLeft and NormRight never adds negation in recursive calls, so case (5) in NormLeft can only be applied for a finite number of times until all negations have been removed. Hence each mutually recursive call terminates in a base case.

Combine all of the above, Procedure 4.3 transform each range-restricted formula into a set of logically equivalent formulae in ANF. ■

Notice that *auxiliary atoms* are introduced to break the right hand sides of implications into binary disjunctions. Inserting an extra auxiliary atom before disjunctions *reduces* the number of branches since the argument of the extra auxiliary atom only contains free variables in the disjunction.

Example 4.4 (ANF Normalization) Let us illustrate Procedure 4.3 by normalizing selected constraints from the employee-department schema in Section 2.1. Applying Procedure 4.3, we can transform the logical constraint $\texttt{manager}(x, y, z) \rightarrow \neg\exists u, v.\texttt{worker}(x, u, v)$ into ANF constraint $\texttt{manager}(x, y, z) \wedge \texttt{worker}(x, u, v) \rightarrow \bot$. Similarly, we can transform the physical constraint $\texttt{emp}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \exists d.\texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z) \leftrightarrow \exists x, y.\texttt{baseemployee}(e, x, y, z)$ into ANF constraints:

$$\{\texttt{emp}(e) \wedge \texttt{emp-num}(e, x) \wedge \texttt{emp-name}(e, y) \wedge \exists d.\texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, z) \rightarrow \exists x.\texttt{AuxA}(x, e, z),$$
$$\texttt{AuxA}(x, e, z) \rightarrow \exists y.\texttt{AuxB}(x, y, e, z),$$
$$\texttt{AuxB}(x, y, e, z) \rightarrow \texttt{baseemployee}(e, x, y, z),$$
$$\texttt{baseemployee}(e, x, y, z) \rightarrow \texttt{emp}(e),$$
$$\texttt{baseemployee}(e, x, y, z) \rightarrow \texttt{emp-num}(e, x),$$
$$\texttt{baseemployee}(e, x, y, z) \rightarrow \texttt{emp-name}(e, y),$$
$$\texttt{baseemployee}(e, x, y, z) \rightarrow \texttt{emp-dept}(e, d), \texttt{baseemployee}(e, x, y, z) \rightarrow \texttt{dept-num}(d, z).\}$$

4.1.2 Inference and Interpolation

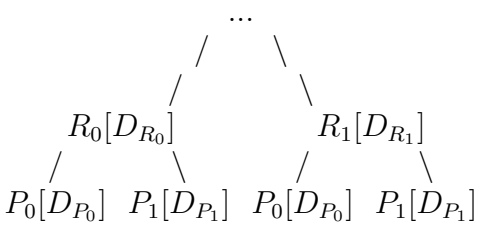
From hereon, we assume all input constraints have been transformed into a set of logically equivalent constraints in ANF. We treat $\bot$ as an atom $\bot()$ with empty arguments. We now proceed to define $\mathfrak{K}$-tableau and its interpolation mechanism. Similar to $\mathfrak{F}$-tableau, $\mathfrak{K}$-tableau is also based on the signed system [19], but instead of explicitly assigning bias labels $\mathsf{L}()/\mathsf{R}()$ to formulae, biases are assigned globally by splitting a single tableau $\mathcal{K}$ into a left and a right tableau $\mathcal{K}^{\mathsf{L}}$ and $\mathcal{K}^{\mathsf{R}}$ such that all L and R biased formulae are stored in $\mathcal{K}^{\mathsf{L}}$ and $\mathcal{K}^{\mathsf{R}}$ respectively. We call $\mathcal{F}^{\overline{\mathsf{X}}}$ the complementary tableau for $\mathcal{F}^{\mathsf{X}}$. The main building blocks of $\mathfrak{K}$-tableau are the so-called *conditional atoms* because a $\mathfrak{K}$-tableau $\mathcal{K}$ is a tuple $(\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ where each $\mathcal{K}^{\mathsf{X}}$ is a set of conditional atoms.

Definition 4.5 (Conditional Atom [32]) *A conditional atom is of the form $r[D]\langle B\rangle$, where r is a ground atom (or $\bot$), $D = \{d_1, \ldots, d_n\}$ is a set of ground* physical atoms *(i.e. ground atoms with predicate symbols $\in \mathsf{S_A}$), and $B = \{i_1 : j_1, \ldots, i_m : j_m\}$ is a set of pairs of numbers $i : j$ (called* branch numbers*) corresponding to a branching point i and a direction j. We call $r[D]\langle B\rangle$ a* conditional atom *for r that depends on ground physical atoms in D (called* dependencies*) and that belongs to tableau* branches *described by the set B.*

$\mathcal{K}^{\mathsf{X}}$ is initialized by a set of conditional atoms with empty dependencies and branches, representing the root of a proof tree. It is important to note that unlike $\mathfrak{F}$-tableau, the tree structure of $\mathfrak{K}$-tableau is implicitly represented by branch numbers and can hence be reconstructed, so it suffices to store the $\mathfrak{K}$-tableau as two sets of conditional atoms. The use of branch numbers allows $\mathfrak{K}$-tableau to apply branch factoring until the tableau construction

phase is complete, which provides a number of practical utilities including reduced memory usage and simplified representation. The set of dependencies for r records the set of all physical atoms required to derive r from the premises, which is crucial for proof closure and interpolation.

Example 4.6 (Branch Factoring) The set of conditional atoms $\{R_0[D_{R_0}]\langle 0:0\rangle, R_1[D_{R_1}]\langle 0:1\rangle, P_0[D_{P_0}]\langle 1:0\rangle, P_1[D_{P_1}]\langle 1:1\rangle$ use branch numbers to factor branches in the following proof tree without branch numbers:



Definition 4.7 ($\mathfrak{K}$-Tableau Inference Rules [25]) *Let $\mathcal{K} = (\mathcal{K}^{\mathsf{X}}, \mathcal{K}^{\overline{\mathsf{X}}})$ denote a $\mathfrak{K}$-tableau, and let $\mathsf{X}(\Sigma)$ denote a set of X biased premises in ANF where $\mathsf{Fv}(\psi \to \varphi)$ are assumed to be universally quantified. For each i, let R, R_i, P, P_i denote predicate symbols, $\bar{x}, \bar{x}_i, \bar{y}$ variables, and $\bar{t}_R, \bar{t}_{R_i}, \bar{t}_P$ ground terms. Let m be a fresh branch number for $\mathcal{K}$ and f^{X} a fresh biased Skolem function symbol for $\exists v.P(v, \bar{y})$. $\mathfrak{K}$-tableau inference rules are defined as follows:*

$\in \mathcal{K}^{\mathsf{X}}$	$\in \mathsf{X}(\Sigma)$	$\cup\, \mathcal{K}^{\mathsf{X}}$
$R(\bar{t}_R)[D]\langle B\rangle$	$R(\bar{x}) \to P(\bar{y})$	$P(\bar{t}_P)[D]\langle B\rangle$
$R(\bar{t}_R)[D]\langle B\rangle$	$R(\bar{x}) \to \exists v.P(v, \bar{y})$	$P(f^{\mathsf{X}}(\bar{t}_P), \bar{t}_P)[D]\langle B\rangle$
$R_1(\bar{t}_{R_1})[D_1]\langle B_1\rangle$ $\vdots$ $R_k(\bar{t}_{R_k})[D_k]\langle B_k\rangle$	$R_1(\bar{x}_1) \wedge \ldots \wedge R_k(\bar{x}_k) \to P(\bar{y})$	$P(\bar{t}_P)[D_1 \cup \cdots \cup D_k]\langle B_1 \cup \cdots \cup B_k\rangle$
$R(\bar{t}_R)[D]\langle B\rangle$	$R(\bar{x}) \to P_1(\bar{y}) \vee P_2(\bar{y})$	$P_1(\bar{t}_P)[D]\langle B \cup \{m:0\}\rangle$ $P_2(\bar{t}_P)[D]\langle B \cup \{m:1\}\rangle$

where $\bar{y} \subseteq \bar{x}$ and $\bar{y} \subseteq \bar{x}_1 \cup \cdots \cup \bar{x}_k$. Constants $\bar{t}_R, \bar{t}_{R_1}, \ldots, \bar{t}_{R_k}$ are matched with universally quantified variables in the left hand side of constraints to derive new conditional atoms with arguments $\bar{t}_P$.

To simplify nested conjunctions, we can apply the same technique used for binarizing nested disjunctions to break nested conjunctions on the left hand sides into binary conjunctions by inserting auxiliary atoms. It remains open whether binary conjunctions are more efficient than nested conjunctions for theorem proving. For the sake of convenience, we hereon assume that all conjunction are binary although this is not necessary in general.

Definition 4.8 ($\mathfrak{K}$-Tableau Physical Rule [25]) *Let P be a physical predicate symbol in $\mathsf{S_A}$ and $\bar{t}$ terms. A* physical rule *for $P(\bar{t})$ inserts a conditional atom $P(\bar{t})[\{P(\bar{t})\}]\langle\rangle$ in $\mathcal{K}^{\mathsf{X}}$.*

Roughly speaking, $P(\bar{t})[\{P(\bar{t})\}]\langle\rangle$ is a result of tautology $\forall\bar{x}.(P(\bar{x}) \rightarrow P(\bar{x}))$. Hence, it is correct to insert $P(\bar{t})[\{P(\bar{t})\}]\langle\rangle$ for arbitrary formula $P(\bar{t})$ in $\mathcal{K}^{\mathsf{X}}$. On the other hand, the freedom to insert arbitrary tautologies is a significant source of non-determinism. We discuss the nuance and significance of choosing appropriate instances of the physical rules in Section 4.3.

Example 4.9 (Conditional Tableau Construction) Consider the user query $q(x, y, z) \leftrightarrow$ `employee`(x, y, z) with respect to the employee-department schema in Section 2.1. For the sake of readability, we use shorthand `EMP` for `employee`, `DEPT` for `department`, `bemp` for `baseemployee`, `enum` for `emp-num`, `ename` for `emp-name`, `edept` for `emp-dept`, `dnum` for `dept-num`, and `efile` for `emp-file`.

We initialize the $\mathfrak{K}$-tableau $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ with L biased query atom $q(0, 1, 2)[]\langle\rangle$ with initial Skolem constants 0, 1, and 2 and biased Skolem terms $t_1 = f^{\mathsf{L}}(0, 1, 2)$ and $t_2 = g^{\mathsf{L}}(f^{\mathsf{L}}(0, 1, 2))$. Then, applying $\mathfrak{K}$-tableau inference rules, we can construct the $\mathfrak{K}$-tableau in Figure 4.1.

Next, we describe the interpolation mechanism of $\mathfrak{K}$-tableau, which can be thought of as a 'meet-in-the-middle' approach:

1. We extract from closed $\mathfrak{K}$-tableau proof $\mathcal{K}$ features CS, called *closing sets*, that are shared among many closed tableau proof trees;
2. We extract from a formula ψ features KS, called *complementary closing sets*, that characterize ψ; and
3. We 'meet in the middle': recognize if ψ is an interpolant by checking the so-called *closing condition* on CS and KS.

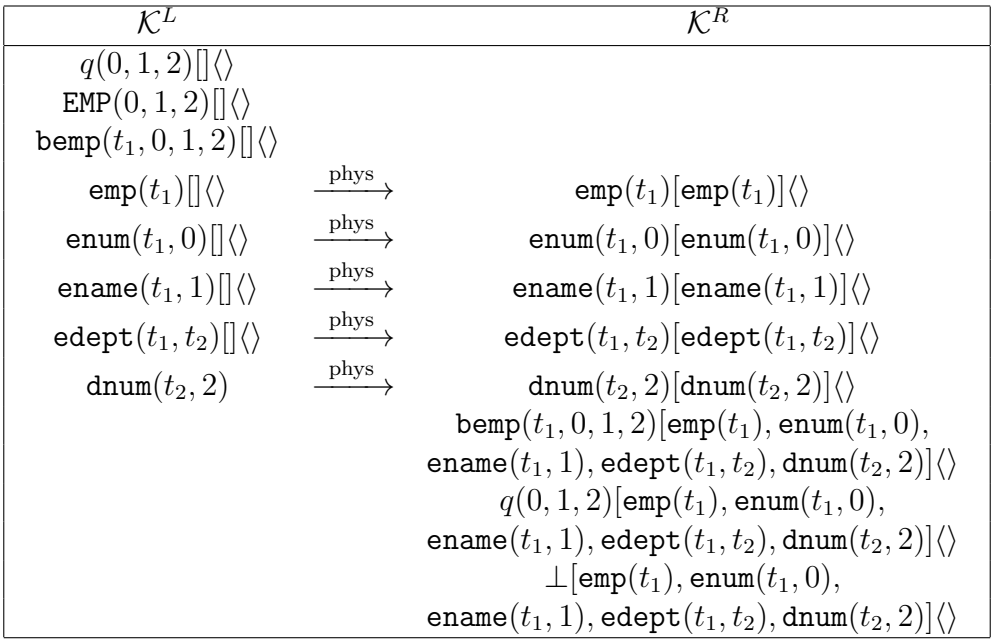


Figure 4.1: A conditional tableau.

We proceed to define closing sets, complementary closing sets, and closing condition in the given order.

Construction 4.10 (Closing Sets [25]) Given a set of conditional atoms $\mathcal{K}^{\mathsf{X}}$, a set of closing sets CS^{X} for $\mathcal{K}^{\mathsf{X}}$ is computed as follows:

1. Initialize $\mathsf{CS}^{\mathsf{X}} := \{\{\neg r, d_1, \ldots, d_k\}\langle B\rangle \mid r[\{d_1, \ldots, d_k\}]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}, r \text{ is physical or } \bot\}$.
2. Repeat steps 3 and 4 for each branching point n from the most to least recent.
3. Let $K = \emptyset$. For each pair of $S_1\langle B_1 \cup \{n : 0\}\rangle, S_2\langle B_2 \cup \{n : 1\}\rangle \in \mathsf{CS}^{\mathsf{X}}$, set $K := K \cup \{S_1 \cup S_2\langle B_1 \cup B_2\rangle\}$ if $\{m : 0, m : 1\} \nsubseteq B_1 \cup B_2$ for some $m \neq n$.
4. $\mathsf{CS}^{\mathsf{X}} := (\mathsf{CS}^{\mathsf{X}} \cup K) - \{S\langle B \cup \{n : i\}\rangle \in \mathsf{CS}^{\mathsf{X}} \mid i \in \{0, 1\}\}$.
5. Remove all $s\langle B\rangle$ from CS^{X} where there exists m such that $|\{m : 0, m : 1\} \cap B| = 1$.
6. Drop $\langle B\rangle$ for all $s\langle B\rangle$ in CS^{X}.

This construction is applied to both the left and right tableaux $\mathcal{K}^{\mathsf{L}}$ and $\mathcal{K}^{\mathsf{R}}$ yielding CS^{L} and CS^{R}.

Example 4.9 (continued) Applying Construction 4.10 to $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$, we get closing sets

$$\mathsf{CS}^{\mathsf{L}} = \{\{\neg\mathtt{emp}(t_1)\}, \{\neg\mathtt{enum}(t_1, 0)\}, \{\neg\mathtt{ename}(t_1, 1)\}, \{\neg\mathtt{edept}(t_1, t_2)\}, \{\neg\mathtt{dnum}(t_2, 2)\}\},$$
$$\mathsf{CS}^{\mathsf{R}} = \{\mathtt{emp}(t_1), \mathtt{enum}(t_1, 0), \mathtt{ename}(t_1, 1), \mathtt{edept}(t_1, t_2), \mathtt{dnum}(t_2, 2)\},$$

where subsumed closing sets have been removed.

Construction 4.11 (Complementary Closing Sets [25]) Given formulae ψ, ψ_1 and ψ_2, the left and right sets of complementary closing sets KS^{L}, KS^{R} for ψ are computed recursively as follows:

ψ :	$\mathsf{KS}^{\mathsf{L}}{}_{\psi}$	$\mathsf{KS}^{\mathsf{R}}{}_{\psi}$
r :	$\{\{\neg r\}\}$	$\{\{r\}\}$
$\psi_1 \wedge \psi_2$:	$\mathsf{KS}^{\mathsf{L}}{}_{\psi_1} \cup \mathsf{KS}^{\mathsf{L}}{}_{\psi_2}$	$\{S_1 \cup S_2 \mid S_1 \in \mathsf{KS}^{\mathsf{R}}{}_{\psi_1}, S_2 \in \mathsf{KS}^{\mathsf{R}}{}_{\psi_2}\}$
$\psi_1 \vee \psi_2$:	$\{S_1 \cup S_2 \mid S_1 \in \mathsf{KS}^{\mathsf{L}}{}_{\psi_1}, S_2 \in \mathsf{KS}^{\mathsf{L}}{}_{\psi_2}\}$	$\mathsf{KS}^{\mathsf{R}}{}_{\psi_1} \cup \mathsf{KS}^{\mathsf{R}}{}_{\psi_2}$
$\neg\psi_1$:	$\mathsf{KS}^{\mathsf{R}}{}_{\psi_1}$	$\mathsf{KS}^{\mathsf{L}}{}_{\psi_1}$
$\exists x.\psi_1$:	$\mathsf{KS}^{\mathsf{R}}{}_{\psi_1[x/t^{\mathsf{X}}]}$	$\mathsf{KS}^{\mathsf{L}}{}_{\psi_1[x/t^{\mathsf{X}}]}$

where t^{X} is an appropriate biased Skolem term, and $\psi = r$ is the base case, where r is an atom.

Example 4.10 (continued) Applying Construction 4.11 to the formula $\psi = \exists e, d.\mathtt{emp}(e) \wedge \mathtt{emp\text{-}num}(e, 0) \wedge \mathtt{emp\text{-}name}(e, 1) \wedge \mathtt{emp\text{-}dept}(e, d) \wedge \mathtt{dept\text{-}num}(d, 3)$, we get complimentary closing sets

$$\mathsf{KS}^{\mathsf{L}} = \{\{\neg\mathtt{emp}(t_1)\}, \{\neg\mathtt{enum}(t_1, 0)\}, \{\neg\mathtt{ename}(t_1, 1)\}, \{\neg\mathtt{edept}(t_1, t_2)\}, \{\neg\mathtt{dnum}(t_2, 2)\}\},$$
$$\mathsf{KS}^{\mathsf{R}} = \{\mathtt{emp}(t_1), \mathtt{enum}(t_1, 0), \mathtt{ename}(t_1, 1), \mathtt{edept}(t_1, t_2), \mathtt{dnum}(t_2, 2)\}.$$

Definition 4.12 (Closing Condition [25]) *Let $\mathcal{K}$ be a conditional tableau for the definability condition in Proposition 2.8 with closing sets $\mathsf{CS}^{\mathsf{L}}, \mathsf{CS}^{\mathsf{R}}$, and let ψ be a formula with complementary closing sets $\mathsf{KS}^{\mathsf{L}}, \mathsf{KS}^{\mathsf{R}}$. We say $\mathcal{K}$ and ψ satisfy the closing condition if for each $s' \in \mathsf{KS}^{\mathsf{L}}(\mathsf{KS}^{\mathsf{R}})$, there exists $s \in \mathsf{CS}^{\mathsf{L}}(\mathsf{CS}^{\mathsf{R}})$ such that $s \subseteq s'$ respectively.*

Finally, the following theorem recognizes an interpolant ψ for user query q with respect to $\Sigma \cup \Sigma_{\mathsf{A}}$ by checking the closing condition on the closing sets for $\mathcal{K}$ and the complementary closing sets for ψ. Consequently, we can use $\mathfrak{K}$-tableau to recognize valid query plans for the purpose of query optimization.

Theorem 4.13 (ℜ-Tableau Interpolation [25]) Let $\mathcal{K}$ be a conditional tableau for the definability condition in Proposition 2.8, and let ψ be a formula. If $\mathcal{K}$ and ψ satisfy the closing condition in Definition 4.12, then $\psi[\bar{0}/\bar{y}]$ is an interpolant for $q^{\mathsf{L}}(\bar{y}) \rightarrow q^{\mathsf{R}}(\bar{y})$.

Example 4.12 (continued) Observe that $\mathsf{CS}^{\mathsf{L}} = \mathsf{KS}^{\mathsf{L}}$ and $\mathsf{CS}^{\mathsf{R}} = \mathsf{KS}^{\mathsf{R}}$, so the closing condition clearly holds. Hence, $\psi = \exists e, d.\texttt{emp}(e) \wedge \texttt{emp-num}(e, 0) \wedge \texttt{emp-name}(e, 1) \wedge \texttt{emp-dept}(e, d) \wedge \texttt{dept-num}(d, 3)$ is a valid query plan for the user query with respect to the input schema.

4.2 Reduction to Free-Variable Tableau

In this section, we prove Theorem 4.13 by reducing ℜ-tableau interpolation to $\mathfrak{F}$-tableau interpolation. The difficulty in establishing such reduction lies in mapping a construction-based mechanism to a decision-based mechanism. $\mathfrak{F}$ tableau interpolation constructs interpolants inductively on the structure of the proof tree based on interpolation rules whereas ℜ-tableau recognizes interpolants among candidate formulae by extracting and comparing features from proofs and formulae. To address this difficulty, we establish two constructions: the first maps a candidate formula ψ to an open $\mathfrak{F}$-tableau; the second maps a ℜ-tableau $\mathcal{K}$ to an open $\mathfrak{F}$-tableau. Then, we attach the open $\mathfrak{F}$-tableaux from the second construction at the tip of the open $\mathfrak{F}$-tableaux from the first construction to create a closed $\mathfrak{F}$-tableau that interpolates to a formula that is logically equivalent to ψ.

The proof roughly proceeds as follows. First, we show that it is possible to construct open $\mathfrak{F}$-tableaux satisfying special closure and interpolation properties. Second, we link the open $\mathfrak{F}$-tableaux into a closed $\mathfrak{F}$-tableau that interpolates to the desired formula.

Definition 4.14 (Attaching Subproof) *Let $\mathcal{F}_1$, $\mathcal{F}_2$ and $\mathcal{F}_3$ be $\mathfrak{F}$-tableaux. We use Fig. 4.2 (a) to denote the tableau obtained by attaching a copy of $\mathcal{F}_2$ at the tip of every left open branch in $\mathcal{F}_1$. We use Fig. 4.2 (b) to denote the tableau obtained by attaching a copy of $\mathcal{F}_3$ at the tip of every right open branch in $\mathcal{F}_1$. We use Fig. 4.2 (c) to denote the tableau obtained by attaching a copy of $\mathcal{F}_2$ at the tip of every left open branch in $\mathcal{F}_1$ and attaching a copy of $\mathcal{F}_3$ at the tip of every right open branch in $\mathcal{F}_1$ simultaneously.*

Definition 4.15 (Complete Subproof) *Let $\mathcal{F}$ be a a $\mathfrak{F}$-tableau proof tree. A complete subproof of $\mathcal{F}$ is a subtree with root node N that contains all descendants of N in $\mathcal{F}$.*

Lemma 4.16 (Interpolating Tableau) Let ψ be a range-restricted formula over S_{A} and $\mathsf{L}(\Sigma_{\mathsf{A}}), \mathsf{R}(\Sigma_{\mathsf{A}})$ premises, where $\Sigma_{\mathsf{A}} = \{\forall \bar{x}.(P^{\mathsf{L}}(\bar{x}) \leftrightarrow P(\bar{x}) \leftrightarrow P^{\mathsf{R}}(\bar{x})) \mid P \in \mathsf{S}_{\mathsf{A}}\}$. Let $\mathsf{KS}^{\mathsf{L}}, \mathsf{KS}^{\mathsf{R}}$ be the complementary closing sets for ψ. For the sake of simplicity, assume all

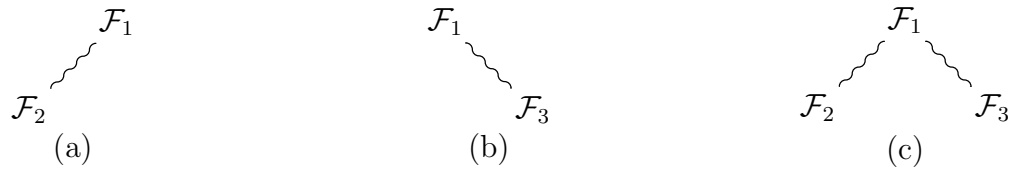


Figure 4.2: Shorthand for attaching subproofs.

quantified variables in ψ are distinct. Let the 'clash' symbol $\otimes$ denote atomic closure on a branch. Then we can construct a $\mathfrak{F}$-tableau $\mathcal{F}_\psi$ called the interpolating tableau for ψ such that the following properties hold:

1. Each left(right) open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}(\mathsf{KS}^{\mathsf{R}})$ that contains all open literals along the branch b, assuming $\mathsf{L}(\mathsf{R})$ superscripts on all atoms in $\mathsf{KS}^{\mathsf{L}}(\mathsf{KS}^{\mathsf{R}})$ respectively.

2. If $\mathcal{F}_\psi$ is a complete subproof of another $\mathfrak{F}$-tableau $\mathcal{F}^*$ and every left(right) open branch in $\mathcal{F}^*$ is atomically closed by a subproof $\mathcal{F}_\alpha(\mathcal{F}_\beta)$ that interpolates to $\alpha(\beta)$ respectively, then the new complete subproof constructed from $\mathcal{F}_\psi$, $\mathcal{F}_\alpha$, and $\mathcal{F}_\beta$ interpolates to a formula ψ' where $\psi' \equiv \alpha \vee (\psi \wedge \beta)$.

proof: We prove by structural induction on ψ and $\neg\psi$ simultaneously.

Base cases: The interpolating tableau $\mathcal{F}_\psi$ for $\psi = R(\bar{x})$ and $\psi = \neg R(\bar{x})$ is shown in Fig. 4.3 (a) and (b) respectively.

By Construction 4.11, the left and right complementary closing sets for $R(\bar{x})$ are $\mathsf{KS}^{\mathsf{L}} = \{\{\neg R(\bar{x})\}\}$ and $\mathsf{KS}^{\mathsf{R}} = \{\{R(\bar{x})\}\}$; and the left and right complementary closing sets for $\neg R(\bar{x})$ are $\mathsf{KS}^{\mathsf{L}} = \{\{R(\bar{x})\}\}$ and $\mathsf{KS}^{\mathsf{R}} = \{\{\neg R(\bar{x})\}\}$. By inspection of the interpolating tableau, there is a single left open branch and a single right open branch in $\mathcal{F}_\psi$, each containing an open literal (marked in red), which corresponds to the complementary closing sets exactly. Hence, property (1) holds for both of the base cases.

Then, construct a new complete subproof from $\mathcal{F}_\psi$ by attaching $\mathcal{F}_\alpha(\mathcal{F}_\beta)$ to each left and right open branch in $\mathcal{F}_\psi$, and we get Fig. 4.4 (a) and (b) respectively.

where intermediate interpolants are marked in blue. Hence by inspection, the resulting tableau interpolates to ψ', where $\psi' \equiv \alpha \vee (R(\bar{x}) \wedge \beta)$ and $\psi' \equiv \alpha \vee (\neg R(\bar{x}) \wedge \beta)$ for $\psi = R(\bar{x})$ and $\psi = \neg R(\bar{x})$ respectively.

Finally, since $\neg(R(\bar{x})) = \neg R(\bar{x})$ and $\neg(\neg R(\bar{x})) = R(\bar{x})$, properties (1) and (2) hold for ψ and $\neg\psi$ simultaneously.

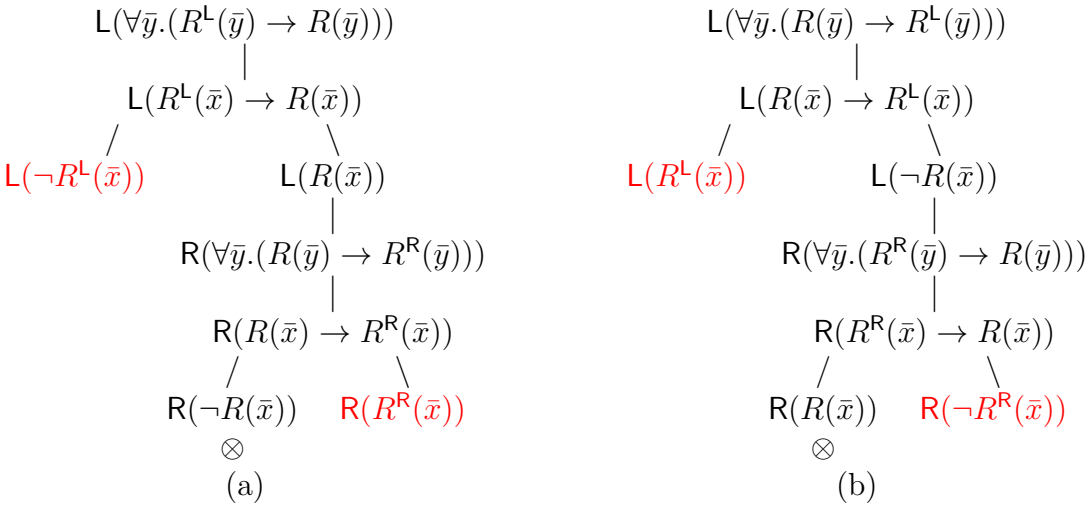


Figure 4.3: Interpolating tableau base cases.

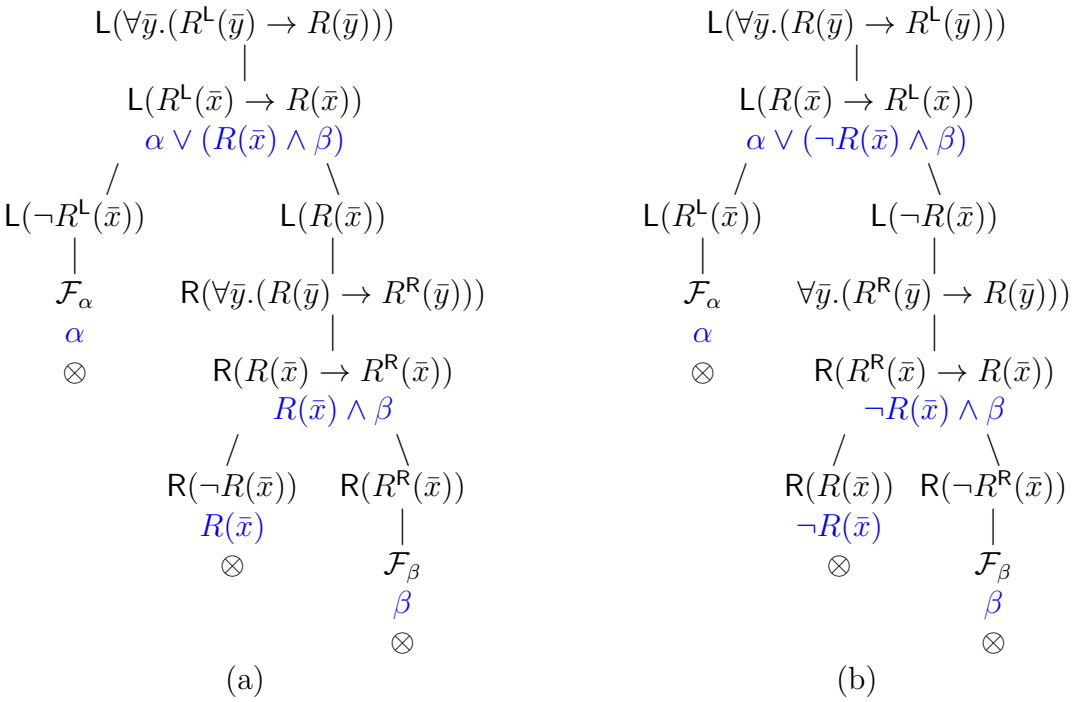


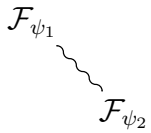
Figure 4.4: Interpolating tableau base cases continued.

Inductive hypothesis: Let ψ_1 and ψ_2 be range-restricted formulae with complementary

closing sets $\mathsf{KS}_1^{\mathsf{L}}, \mathsf{KS}_1^{\mathsf{R}}$ and $\mathsf{KS}_2^{\mathsf{L}}, \mathsf{KS}_2^{\mathsf{R}}$ respectively. Let $\neg\psi_1$ and $\neg\psi_2$ be range-restricted formulae with complementary closing sets $\mathsf{KS}_1^{\mathsf{R}}, \mathsf{KS}_1^{\mathsf{L}}$ and $\mathsf{KS}_2^{\mathsf{R}}, \mathsf{KS}_2^{\mathsf{L}}$ respectively. Suppose we can construct interpolating tableaux $\mathcal{F}_{\psi_1}$ and $\mathcal{F}_{\psi_2}$ for ψ_1 and ψ_2 respectively such that properties (1) and (2) hold. Similarly, suppose we can construct interpolating tableaux $\mathcal{F}_{\neg\psi_1}$ and $\mathcal{F}_{\neg\psi_2}$ for $\neg\psi_1$ and $\neg\psi_2$ respectively such that properties (1) and (2) hold.

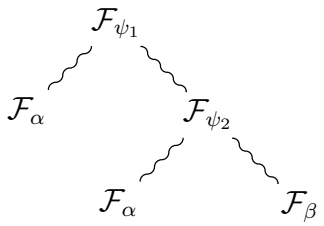
Inductive proof: We prove that we can construct interpolating tableau $\mathcal{F}_\psi$ for ψ and $\mathcal{F}_{\neg\psi}$ for $\neg\psi$, where ψ is any of $\psi_1 \wedge \psi_2$, $\psi_1 \vee \psi_2$ and $\exists x.\psi_1$, such that properties (1) and (2) hold.

Conjunctive cases: We construct the interpolating tableau $\mathcal{F}_\psi$ for $\psi = \psi_1 \wedge \psi_2$ by attaching $\mathcal{F}_{\psi_2}$ at the tip of every right open branch in $\mathcal{F}_{\psi_1}$. Pictorially, we have



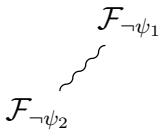
By the inductive hypothesis, each left open branch b in $\mathcal{F}_{\psi_1}$ ($\mathcal{F}_{\psi_2}$) corresponds to a set $s_1 \in \mathsf{KS}_1^{\mathsf{L}}(s_2 \in \mathsf{KS}_2^{\mathsf{L}})$ that contains all open literals along b respectively. Similarly, each right open branch b in $\mathcal{F}_{\psi_1}$ ($\mathcal{F}_{\psi_2}$) corresponds to a set $s_1 \in \mathsf{KS}_1^{\mathsf{R}}(s_2 \in \mathsf{KS}_2^{\mathsf{R}})$ that contains all open literals along b respectively. Since $\mathcal{F}_2$ is attached at the tip of every right open branch in $\mathcal{F}_{\psi_1}$, the set of left open branches in $\mathcal{F}_\psi$ is the union of left open branches in $\mathcal{F}_{\psi_1}$ and left open branches in $\mathcal{F}_{\psi_2}$. Then since $\mathsf{KS}_1^{\mathsf{L}} \cup \mathsf{KS}_2^{\mathsf{L}} = \mathsf{KS}^{\mathsf{L}}$ by Construction 4.11, every left open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}$ that contains all open literals along b. Since $\mathcal{F}_{\psi_2}$ is attached at the tip of every right open branch in $\mathcal{F}_{\psi_1}$, the set of right open branches in $\mathcal{F}_\psi$ is the concatenation of each right open branch in $\mathcal{F}_{\psi_1}$ and each right open branch in $\mathcal{F}_{\psi_2}$. Then since $\mathsf{KS}_1^{\mathsf{R}} \times \mathsf{KS}_2^{\mathsf{R}} = \mathsf{KS}^{\mathsf{R}}$ by Construction 4.11, every right open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{R}}$ that contains all open literals along b. Hence, property (1) holds.

Then, consider the closed tableau constructed by attaching $\mathcal{F}_\alpha$ to every left open branch and $\mathcal{F}_\beta$ to every right open branch in $\mathcal{F}_\psi$. Pictorially, we have



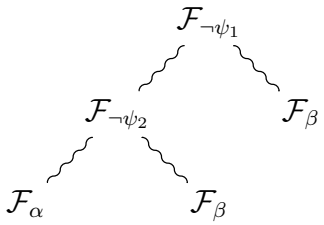
By the inductive hypothesis, the right subproof consisting of $\mathcal{F}_{\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to ψ_2', where $\psi_2' \equiv (\psi_2 \wedge \beta) \vee \alpha$, and the complete subproof consisting of $\mathcal{F}_{\psi_1}$, $\mathcal{F}_{\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to ψ', where $\psi' \equiv (\psi_1 \wedge \psi_2') \vee \alpha \equiv ((\psi_1 \wedge \psi_2) \wedge \beta) \vee \alpha$. Hence, property (2) holds.

Negative conjunctive cases: consider $\neg\psi \equiv \neg(\psi_1 \wedge \psi_2) \equiv \neg\psi_1 \vee \neg\psi_2$. By the inductive hypothesis, we can construct interpolating tableaux $\mathcal{F}_{\neg\psi_1}$ and $\mathcal{F}_{\neg\psi_2}$ for $\neg\psi_1$ and $\neg\psi_2$ respectively such that properties (1) and (2) hold. Then, we can construct interpolating tableau $\mathcal{F}_{\neg\psi}$ for $\neg\psi$ by attaching $\mathcal{F}_{\neg\psi_2}$ to every left open branch of $\mathcal{F}_{\neg\psi_1}$. Pictorially, we have



By Construction 4.11, the left complementary closing sets for $\neg\psi_1$, $\neg\psi_2$ and $\neg\psi$ are $\mathsf{KS}_1^{\mathsf{R}}$, $\mathsf{KS}_2^{\mathsf{R}}$ and KS^{R} respectively. Then since a copy of $\mathcal{F}_{\neg\psi_2}$ is attached at the tip of every left open branch in $\mathcal{F}_{\neg\psi_1}$, the set of left open branches in $\mathcal{F}_{\neg\psi}$ is the concatenation of each left open branch in $\mathcal{F}_{\neg\psi_1}$ and each left open branch in $\mathcal{F}_{\neg\psi_2}$. Hence, since $\mathsf{KS}_1^{\mathsf{R}} \times \mathsf{KS}_2^{\mathsf{R}} = \mathsf{KS}^{\mathsf{R}}$, every left open branch b in $\mathcal{F}_{\neg\psi}$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{R}}$ that contains all open literals along b. Similarly, by Construction 4.11, the right complementary closing sets for $\neg\psi_1$, $\neg\psi_2$ and $\neg\psi$ are $\mathsf{KS}_1^{\mathsf{L}}$, $\mathsf{KS}_2^{\mathsf{L}}$ and KS^{L} respectively. Then since a copy of $\mathcal{F}_{\neg\psi_2}$ is attached at the tip of every left open branch in $\mathcal{F}_{\neg\psi_1}$, the set of right open branches in $\mathcal{F}_{\neg\psi}$ is the union of right open branches in $\mathcal{F}_{\neg\psi_1}$ and right open branches in $\mathcal{F}_{\neg\psi_2}$. Hence, since $\mathsf{KS}_1^{\mathsf{L}} \cup \mathsf{KS}_2^{\mathsf{L}} = \mathsf{KS}^{\mathsf{L}}$, every right open branch b in $\mathcal{F}_{\neg\psi}$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}$ that contains all open literals along b. Thus, property (1) holds for $\neg\psi$.

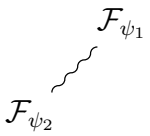
Then, consider the closed tableau constructed by attaching $\mathcal{F}_\alpha$ to every left open branch and $\mathcal{F}_\beta$ to every right open branch in $\mathcal{F}_{\neg\psi}$. Pictorially, we have



By the inductive hypothesis, the left subproof consisting of $\mathcal{F}_{\neg\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to $\neg\psi_2'$, where $\neg\psi_2' \equiv (\neg\psi_2 \wedge \beta) \vee \alpha$, and the complete subproof consisting of $\mathcal{F}_{\neg\psi_1}$, $\mathcal{F}_{\neg\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to $\neg\psi'$, where $\neg\psi' \equiv (\neg\psi_1 \wedge \beta) \vee \neg\psi_2' \equiv (\neg(\psi_1 \wedge \psi_2) \wedge \beta) \vee \alpha$. Hence, property (2) holds.

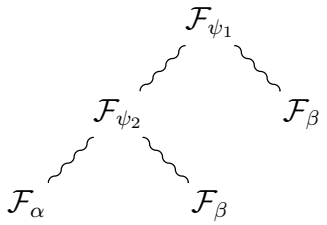
Combining the above, both properties hold for conjunctive cases and their negations simultaneously.

Disjunctive cases: We construct the interpolating tableau $\mathcal{F}$ for $\psi = \psi_1 \vee \psi_2$ by attaching $\mathcal{F}_{\psi_2}$ at the tip of every left open branch in $\mathcal{F}_{\psi_1}$. Pictorially, we have



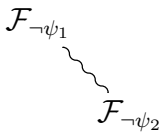
By the inductive hypothesis, each left open branch b in $\mathcal{F}_{\psi_1}$ ($\mathcal{F}_{\psi_2}$) corresponds to a set $s_1 \in \mathsf{KS}_1^{\mathsf{L}}$($s_2 \in \mathsf{KS}_2^{\mathsf{L}}$) that contains all open literals along b respectively. Similarly, each right open branch b in $\mathcal{F}_{\psi_1}$ ($\mathcal{F}_{\psi_2}$) corresponds to a set $s_1 \in \mathsf{KS}_1^{\mathsf{R}}$($s_2 \in \mathsf{KS}_2^{\mathsf{R}}$) that contains all open literals along b respectively. Since $\mathcal{F}_{\psi_2}$ is attached at the tip of every left open branch in $\mathcal{F}_{\psi_1}$, the set of right open branches in $\mathcal{F}_\psi$ is the union of right open branches in $\mathcal{F}_{\psi_1}$ and right open branches in $\mathcal{F}_{\psi_2}$. Then since $\mathsf{KS}_1^{\mathsf{R}} \cup \mathsf{KS}_2^{\mathsf{R}} = \mathsf{KS}^{\mathsf{R}}$ by Construction 4.11, every right open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{R}}$ that contains all open literals along b. Since $\mathcal{F}_{\psi_2}$ is attached at the tip of every left open branch in $\mathcal{F}_{\psi_1}$, the set of left open branches in $\mathcal{F}_\psi$ is the concatenation of each left open branch in $\mathcal{F}_{\psi_1}$ and each left open branch in $\mathcal{F}_{\psi_2}$. Then since $\mathsf{KS}_1^{\mathsf{L}} \times \mathsf{KS}_2^{\mathsf{L}} = \mathsf{KS}^{\mathsf{L}}$ by Construction 4.11, every left open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}$ that contains all open literals along b. Hence, property (1) holds.

Then, consider the closed tableau constructed by attaching $\mathcal{F}_\alpha$ to every left open branch and $\mathcal{F}_\beta$ to every right open branch in $\mathcal{F}_\psi$. Pictorially, we have



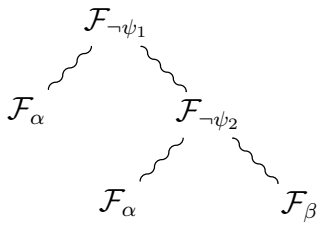
By the inductive hypothesis, the left subproof consisting of $\mathcal{F}_{\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to ψ_2', where $\psi_2' \equiv (\psi_2 \wedge \beta) \vee \alpha$, and the complete subproof consisting of $\mathcal{F}_{\psi_1}$, $\mathcal{F}_{\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to ψ', where $\psi' \equiv (\psi_1 \wedge \beta) \vee \psi_2' \equiv ((\psi_1 \vee \psi_2) \wedge \beta) \vee \alpha$. Hence, property (2) holds.

Negative disjunctive cases: Consider $\neg\psi \equiv \neg(\psi_1 \vee \psi_2) \equiv \neg\psi_1 \wedge \neg\psi_2$. By the inductive hypothesis, we can construct interpolating tableaux $\mathcal{F}_{\neg\psi_1}$ and $\mathcal{F}_{\neg\psi_2}$ for $\neg\psi_1$ and $\neg\psi_2$ respectively such that properties (1) and (2) hold. Then, we can construct interpolating tableau $\mathcal{F}_{\neg\psi}$ for $\neg\psi$ by attaching $\mathcal{F}_{\neg\psi_2}$ to every right open branch of $\mathcal{F}_{\neg\psi_1}$. Pictorially, we have



By Construction 4.11, the right complementary closing sets for $\neg\psi_1$, $\neg\psi_2$ and $\neg\psi$ are $\mathsf{KS}_1^{\mathsf{L}}$, $\mathsf{KS}_2^{\mathsf{L}}$ and KS^{L} respectively. Then since a copy of $\mathcal{F}_{\neg\psi_2}$ is attached at the tip of every right open branch in $\mathcal{F}_{\neg\psi_1}$, the set of right open branches in $\mathcal{F}_{\neg\psi}$ is the concatenation of each right open branch in $\mathcal{F}_{\neg\psi_1}$ and each right open branch in $\mathcal{F}_{\neg\psi_2}$. Hence, since $\mathsf{KS}_1^{\mathsf{L}} \times \mathsf{KS}_2^{\mathsf{L}} = \mathsf{KS}^{\mathsf{L}}$, every right open branch b in $\mathcal{F}_{\neg\psi}$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}$ that contains all open literals along b. Similarly, by Construction 4.11, the left complementary closing sets for $\neg\psi_1$, $\neg\psi_2$ and $\neg\psi$ are $\mathsf{KS}_1^{\mathsf{R}}$, $\mathsf{KS}_2^{\mathsf{R}}$ and KS^{R} respectively. Then since a copy of $\mathcal{F}_{\neg\psi_2}$ is attached at the tip of every right open branch in $\mathcal{F}_{\neg\psi_1}$, the set of left open branches in $\mathcal{F}_{\neg\psi}$ is the union of left open branches in $\mathcal{F}_{\neg\psi_1}$ and left open branches in $\mathcal{F}_{\neg\psi_2}$. Hence, since $\mathsf{KS}_1^{\mathsf{R}} \cup \mathsf{KS}_2^{\mathsf{R}} = \mathsf{KS}^{\mathsf{R}}$, every left open branch b in $\mathcal{F}_{\neg\psi}$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{R}}$ that contains all open literals along b. Thus, property (1) holds for $\neg\psi$.

Then, consider the closed tableau constructed by attaching $\mathcal{F}_\alpha$ to every left open branch and $\mathcal{F}_\beta$ to every right open branch in $\mathcal{F}_{\neg\psi}$. Pictorially, we have



By the inductive hypothesis, the right subproof consisting of $\mathcal{F}_{\neg\psi_2}$, $\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to $\neg\psi_2'$, where $\neg\psi_2' \equiv (\neg\psi_2 \wedge \beta) \vee \alpha$, and the complete subproof consisting of $\mathcal{F}_{\neg\psi_1}$, $\mathcal{F}_{\neg\psi_2}$,

$\mathcal{F}_\alpha$ and $\mathcal{F}_\beta$ interpolates to $\neg\psi'$, where $\neg\psi' \equiv (\neg\psi_1 \wedge \neg\psi_2') \vee \alpha \equiv (\neg(\psi_1 \vee \psi_2) \wedge \beta) \vee \alpha$. Hence, property (2) holds.

Combining the above, both properties hold for all disjunctive cases and their negations.

Existential cases: We construct the interpolating tableau $\mathcal{F}_\psi$ for $\psi = \exists x.\psi_1$ by substituting every occurrence of x in $\mathcal{F}_{\psi_1}$ with biased Skolem term $f^{\mathsf{L}}(\mathsf{Fv}(\psi))$. By the inductive hypothesis, each left (right) open branch b in $\mathcal{F}_{\psi_1}$ corresponds to a set $s \in \mathsf{KS}_1^{\mathsf{L}}(s \in \mathsf{KS}_1^{\mathsf{R}})$ that contains all open literals along b respectively. Then since $\mathsf{KS}^{\mathsf{L}} = \mathsf{KS}_1^{\mathsf{L}}[x/f^{\mathsf{L}}(\mathsf{Fv}(\psi))]$ and $\mathsf{KS}^{\mathsf{R}} = \mathsf{KS}_1^{\mathsf{R}}[x/f^{\mathsf{L}}(\mathsf{Fv}(\psi))]$, each left (right) open branch b in $\mathcal{F}_\psi$ corresponds to a set $s \in \mathsf{KS}^{\mathsf{L}}(s \in \mathsf{KS}^{\mathsf{R}})$ that contains all open literals along b respectively. Hence, property (1) holds for ψ.

By the inductive hypothesis, $\mathcal{F}_{\psi_1}$ interpolates to $\psi_1' \equiv \psi_1$. Then since $\mathcal{F}_\psi$ is constructed by replacing every x in $\mathcal{F}_{\psi_1}$ by $f^{\mathsf{L}}(\mathsf{Fv}(\psi))$, $\mathcal{F}_\psi$ interpolates to $\psi_1'[x/f^{\mathsf{L}}(\mathsf{Fv}(\psi))]$. However, since $f^{\mathsf{L}}(\mathsf{Fv}(\psi))$ is left biased and contains free variables only, we can apply quantification rule (Q1) to quantify $f^{\mathsf{L}}(\mathsf{Fv}(\psi))$ in $\psi_1'[x/f^{\mathsf{L}}(\mathsf{Fv}(\psi))]$, so $\mathcal{F}_\psi$ interpolates to $\exists x.\psi_1' \equiv= \psi$. Hence, property (2) holds for ψ.

Negated Existential cases: We construct the interpolating tableau $\mathcal{F}_{\neg\psi}$ for $\neg\psi = \neg(\exists x.\psi_1)$ by substituting every occurrence of x in $\mathcal{F}_{\neg\psi_1}$ with biased Skolem term $f^{\mathsf{R}}(\mathsf{Fv}(\neg\psi))$. Property (1) and (2) follow from the fact that $\neg(\exists x.\psi_1) \equiv \forall x.(\neg\psi_1)$ and right biased Skolem term $f^{\mathsf{R}}(\mathsf{Fv}(\neg\psi))$ is universally quantified by quantification rule (Q2).

Combining the above, both properties hold for all existential cases and their negations.

This covers all the cases. Thus, we can construct interpolating tableau satisfying properties (1), (2) for all range-restricted formulae. ■

Note that since all quantifiers in $\mathsf{L}(\Sigma_{\mathsf{A}}), \mathsf{R}(\Sigma_{\mathsf{A}})$ are universal, on-the-spot substitution in the proof of Lemma 4.16 is equivalent to applying universal expansions using fresh variables then applying closing substitutions in the end. The resulting proof tree is therefore a valid closed $\mathfrak{F}$-tableau that can be obtained by postponed substitution.

For the proof of correctness, we assume that derivations of the conditional atoms in $\mathcal{K}^{\mathsf{X}}$ are totally ordered although it is sufficient to keep $\mathcal{K}^{\mathsf{X}}$ as an unordered set for practical implementations.

Construction 4.17 (Closing Tableau) A closing tableau $\mathcal{F}$ for $\mathfrak{K}$-tableau $\mathcal{K}$ is an open $\mathfrak{F}$-tableau defined with respect to (i) a function ω, which maps each $\mathfrak{K}$-tableau inference in $\mathcal{K}$ to a sequence of $\mathfrak{F}$-tableau inferences in $\mathcal{F}$, and (ii) a substitution σ. Given a sequence of $\mathfrak{K}$-tableau inferences $(\iota_1, \ldots, \iota_n)$ that construct $\mathcal{K}$, the closing tableau $\mathcal{F}$ for $\mathcal{K}$ is constructed by a sequence of $\mathfrak{F}$-tableau inferences $(\omega(\iota_1), \ldots, \omega(\iota_n))$ under substitution σ.

Each inference $\omega(\iota_i)$ is applied at the tip of selected branches in the tableau constructed from $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$.

The definition of branch numbers for closing tableaux is adapted accordingly. We say that a set of branch numbers B_1 is *associated* with a set of branch numbers B_2 if $B_1 \supseteq B_2$. As a consequence of this adapted notion of branch numbers for closing tableaux, each inference mapping is attached to the proof many times at each step because a copy of $\omega(\iota_i)$ is attached at the tip of all appropriate (i.e., associated) branches.

First, we define the base case for closing tableau construction.

0. If $\mathcal{K}$ is initialized with a set of conditional atoms $\mathcal{I}^{\mathsf{X}}$, then its closing tableau $\mathcal{F}$ is initialized by attaching the same set of biased initial atoms, without conditions and branches, to its root.

Assume all conjunctions in the input constraints are binary. We define ω and σ for the physical rule and each $\mathfrak{K}$-tableau inference rule in Definition 4.7 as follows:

1. Let ι_i be a $\mathfrak{K}$-tableau inference that takes conditional atom $R(\bar{t}_R)[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}$ and a constraint $R(\bar{x}) \to P(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P(\bar{t}_P)[D]\langle B\rangle$ in $\mathcal{K}^{\mathsf{X}}$. We define $\omega(\iota_i) =$

$$\begin{array}{c} \mathsf{X}(\forall \bar{x}.(R(\bar{x}) \to P(\bar{y}))) \\ | \\ \mathsf{X}(R(\bar{v}_x) \to P(\bar{v}_y)) \\ / \qquad \backslash \\ \mathsf{X}(\neg R(\bar{v}_x)) \quad \mathsf{X}(P(\bar{v}_y)) \\ | \\ \mathsf{X}(R(\bar{t}_R)) \\ \otimes \\ \text{closed by } \sigma \end{array}$$

   and $\sigma(\bar{v}_x) = \bar{t}_R$ for fresh variables $\bar{v}_x$. We apply $\omega(\iota)$ to $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$ by attaching a copy of $\omega(\iota_i)$ at the tip of every open branch associated with B in the $\mathfrak{F}$-tableau constructed by $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$. The set of branch numbers for each extended branch is $B \cup B'$ where B' is the set of branch numbers assigned to the branch before extension by $\omega(\iota_i)$.

2. Let ι_i be a $\mathfrak{K}$-tableau inference that takes conditional atoms $R_1(\bar{t}_{R_1})[D_1]\langle B_1\rangle \in \mathcal{K}^{\mathsf{X}}$, $R_2(\bar{t}_{R_2})[D_2]\langle B_2\rangle \in \mathcal{K}^{\mathsf{X}}$ and a constraint $R_1(\bar{x}_1) \wedge R_2(\bar{x}_2) \rightarrow P(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P(\bar{t}_P)[D_1 \cup D_2]\langle B_1 \cup B_2\rangle$ in $\mathcal{K}^{\mathsf{X}}$. We define $\omega(\iota_i) =$

$$\begin{array}{c}
\mathsf{X}(\forall\bar{x}.(R_1(\bar{x}_1) \wedge R_2(\bar{x}_2) \rightarrow P(\bar{y}))) \\
\vert \\
\mathsf{X}(R_1(\bar{v}_{x_1}) \wedge R_2(\bar{v}_{x_2}) \rightarrow P(\bar{v}_y)) \\
/ \qquad\qquad \backslash \\
\mathsf{X}(\neg(R_1(\bar{v}_{x_1}) \wedge R_2(\bar{v}_{x_2}))) \quad \mathsf{X}(P(\bar{v}_y)) \\
/ \qquad \backslash \\
\mathsf{X}(\neg R_1(\bar{v}_{x_1})) \quad \mathsf{X}(\neg R_2(\bar{v}_{x_2})) \\
\vert \qquad\quad \vert \\
\mathsf{X}(R_1(\bar{t}_{R_1})) \quad \mathsf{X}(R_2(\bar{t}_{R_2})) \\
\otimes \qquad\quad \otimes \\
\text{closed by } \sigma \quad \text{closed by } \sigma
\end{array}$$

and $\sigma(\bar{v}_{x_1}) = \bar{t}_{R_1}$ and $\sigma(\bar{v}_{x_2}) = \bar{t}_{R_2}$ for fresh variables $\bar{v}_{x_1}, \bar{v}_{x_2}$. We apply $\omega(\iota_i)$ to $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$ by attaching a copy of $\omega(\iota_i)$ at the tip of every open branch associated with $B_1 \cup B_2$ in the $\mathfrak{F}$-tableau constructed by $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$. The set of branch numbers for each extended branch is $B \cup B'$ where B' is the set of branch numbers assigned to the branch before extension by $\omega(\iota_i)$.

3. Let ι_i be a $\mathfrak{K}$-tableau inference that takes conditional atom $R(\bar{t}_R)[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}$ and a constraint $R(\bar{x}) \rightarrow P_1(\bar{y}) \vee P_2(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atoms $P_1(\bar{t}_P)[D]\langle B \cup \{m:0\}\rangle$ and $P_2(\bar{t}_P)[D]\langle B \cup \{m:1\}\rangle$ in $\mathcal{K}^{\mathsf{X}}$ for a fresh branch number m. We define $\omega(\iota_i) =$

$$\begin{array}{c}
\mathsf{X}(\forall\bar{x}.(R(\bar{x}) \rightarrow P_1(\bar{y}) \vee P_2(\bar{y}))) \\
\vert \\
\mathsf{X}(R(\bar{v}_x) \rightarrow P_1(\bar{v}_y) \vee P_2(\bar{v}_y)) \\
/ \qquad\qquad \backslash \\
\mathsf{X}(\neg R(\bar{v}_x)) \quad \mathsf{X}(P_1(\bar{v}_y) \vee P_2(\bar{v}_y)) \\
\vert \qquad\quad / \qquad \backslash \\
\mathsf{X}(R(\bar{t}_R)) \quad \mathsf{X}(P_1(\bar{v}_y)) \quad \mathsf{X}(P_2(\bar{v}_y)) \\
\otimes \\
\text{closed by } \sigma
\end{array}$$

and $\sigma(\bar{v}_x) = \bar{t}_R$ for fresh variables $\bar{v}_x$. We apply $\omega(\iota_i)$ to $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$ by attaching a copy of $\omega(\iota_i)$ at the tip of every open branch associated with B in the

$\mathfrak{F}$-tableau constructed by $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$. The set of branch numbers for the left(right) extended branch is $B \cup \{m : 0\} \cup B'(B \cup \{m : 1\} \cup B')$ respectively, where B' is the set of branch numbers assigned to the branch before extension by $\omega(\iota_i)$.

4. Let ι_i be a $\mathfrak{K}$-tableau inference that takes conditional atom $R(\bar{t}_R)[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}$ and a constraint $R(\bar{x}) \rightarrow \exists v.P(v, \bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P(f^{\mathsf{X}}(\bar{t}_P), \bar{t}_P)[D]\langle B\rangle$ in $\mathcal{K}^{\mathsf{X}}$, where f^{X} is a X biased Skolem function symbol for $\exists v.P(v, \bar{y})$. We define $\omega(\iota_i) =$

$$\begin{array}{c} \mathsf{X}(\forall \bar{x}.(R(\bar{x}) \rightarrow \exists v.P(v, \bar{y}))) \\ | \\ \mathsf{X}(R(\bar{v}_x) \rightarrow \exists v.P(v, \bar{v}_y)) \\ / \qquad \backslash \\ \mathsf{X}(\neg R(\bar{v}_x)) \quad \mathsf{X}(\exists v.P(v, \bar{v}_y)) \\ | \qquad | \\ \mathsf{X}(R(\bar{t}_R)) \quad \mathsf{X}(P(f^{\mathsf{X}}(\bar{v}_y), \bar{v}_y)) \\ \otimes \\ \text{closed by } \sigma \end{array}$$

and $\sigma(\bar{v}_x) = \bar{t}_R$ for fresh variables $\bar{v}_x$. We apply $\omega(\iota_i)$ to $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$ by attaching a copy of $\omega(\iota_i)$ at the tip of every open branch associated with B in the $\mathfrak{F}$-tableau constructed by $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$. The set of branch numbers for each extended branch is $B \cup B'$ where B' is the set of branch numbers assigned to the branch before extension by $\omega(\iota_i)$.

5. Let ι_i be an application of physical rule which inserts $P(\bar{t}_R)[P(\bar{t}_R)]\langle\rangle$ in $\mathcal{K}^{\mathsf{X}}$. We define $\omega(\iota_i) =$

$$\begin{array}{c} \mathsf{X}(\forall \bar{x}.(P(\bar{x}) \rightarrow P(\bar{x}))) \\ | \\ \mathsf{X}(P(\bar{v}_x) \rightarrow P(\bar{v}_x)) \\ / \qquad \backslash \\ \mathsf{X}(\neg P(\bar{v}_x)) \quad \mathsf{X}(P(\bar{v}_x)) \end{array}$$

and $\sigma(\bar{v}_x) = \bar{t}_R$ for fresh variables $\bar{v}_x$. We apply $\omega(\iota_i)$ to $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$ by attaching a copy of $\omega(\iota_i)$ at the tip of every open branch in the $\mathfrak{F}$-tableau constructed by $(\omega(\iota_1), \ldots, \omega(\iota_{i-1}))$. The set of branch numbers for each extended branch inherits the set of branch numbers assigned to the branch before extension by $\omega(\iota_i)$.

To simplify notation in the proof of correctness, let us specify terminology for special branches and atoms in the closing tableaux resulting from mapping ω and substitution σ.

Definition 4.18 (Major and Minor Branches) *We call branches in $\mathcal{F}$ that result strictly from the mapping ω* minor branches *and those resulting from $\mathfrak{K}$-tableau disjunctive inference rules* major branches*.*

It is clear from construction that every open minor branch contains a single open literal at its leaf node.

Definition 4.19 (Minimal Derivation) *We call a sequence of inferences $(\iota_1 \ldots, \iota_n)$ that derives $r[D]\langle B\rangle$* minimal *if each inference ι_i is necessary for deriving $r[D]\langle B\rangle$.*

Definition 4.20 (Image) *We say an occurrence of an atom r in the closing tableau is an image of $r[D]\langle B\rangle$ derived from $(\iota_1 \ldots, \iota_n)$ if r is in $\omega(\iota_n)\sigma$.*

We say that a minor branch is necessary to derive *the image r for $r[D]\langle B\rangle$ if the minor branch is introduced by $\omega(\iota_j)\sigma$ such that ι_j is in a minimal sequence of derivations $(\iota_1, \ldots, \iota_n)$ for $r[D]\langle B\rangle$.*

The following lemma illustrates the utility of dependencies in conditional atoms. Intuitively, dependencies for an atom r record a necessary and sufficient set of open minor branches that derive r. When dependencies are merged via closing set construction based on branch numbers, extraneous derivations are removed to produce a closing tableau that can be closed by attaching a closing set to its root.

Lemma 4.21 Let $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ be a $\mathfrak{K}$-tableau, and $\mathsf{CS}^{\mathsf{L}}, \mathsf{CS}^{\mathsf{R}}$ its left and right closing sets. Let $\mathcal{I}^{\mathsf{X}}$ be the set of initial atoms for $\mathcal{K}^{\mathsf{X}}$ and $(\iota_1, \ldots, \iota_k)$ be an ordered list of inferences that constructs $\mathcal{K}^{\mathsf{X}}$. Construct closing tableau $\mathcal{F}$ for $\mathcal{K}^{\mathsf{X}}$ from $(\iota_1, \ldots, \iota_k)$ and $\mathcal{I}^{\mathsf{X}}$. Then for each $r[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}$, D contains the negation of the open literal on each minor branch that is necessary to derive the image r in $\mathcal{F}$.

proof: We prove by induction on the number of inferences n.

Case 0 (base case): Consider the base case in which $\mathcal{K}^{\mathsf{X}} = \mathcal{I}^{\mathsf{X}}$ is the initial tableau. By $\mathfrak{F}$-tableau initialization, each conditional atom in $\mathcal{I}^{\mathsf{X}}$ has empty dependencies and branches. Then since there does not exist any minor branches at the root of $\mathcal{F}$, statement holds.

Inductive hypothesis: Let $(\iota_1, \ldots, \iota_n)$ be a sequence of $\mathfrak{K}$-tableau inferences that constructs $\mathcal{K}^{\mathsf{X}}_n$ from initial atoms $\mathcal{I}^{\mathsf{X}}$. Let $\mathcal{F}_n$ be its closing tableau. Assume statement holds for $\mathcal{K}^{\mathsf{X}}_n$.

Inductive proof: Construct $\mathcal{K}^{\mathsf{X}}_{n+1}$ by applying inference ι_{n+1} to $\mathcal{K}^{\mathsf{X}}_{n}$. That is, $\mathcal{K}^{\mathsf{X}}_{n+1}$ is constructed by applying $(\iota_1, \ldots, \iota_{n+1})$ to the set of initial atoms $\mathcal{I}^{\mathsf{X}}$. We prove that statement holds for $\mathcal{K}^{\mathsf{X}}_{n+1}$.

Case 1 ($R(\bar{x}) \rightarrow P(\bar{y})$): Suppose inference ι_{n+1} takes conditional atom $R(\bar{t}_R)[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}_{n}$ and applies constraint $R(\bar{x}) \rightarrow P(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P(\bar{t}_P)[D]\langle B\rangle$ in $\mathcal{K}^{\mathsf{X}}_{n+1}$. By Construction 4.17, the closing tableau $\mathcal{F}_{n+1}$ for $\mathcal{K}^{\mathsf{X}}_{n+1}$ is obtained by attaching a copy of $\omega(\iota_{n+1})\sigma$ at the tip of each branch associated with branch number B in $\mathcal{F}_n$. Then each expansion results in a new left closed branch with atomic closure $\neg R(\bar{v}_x), R(\bar{t}_R)$ under $\bar{v}_x\sigma = \bar{t}_R$ and a new right open branch with open atom $P(\bar{t}_P)$ at its leaf node. In other words, attaching $\omega(\iota_{n+1})\sigma$ at the tip of a branch extends that tip with open atom $P(\bar{t}_P)$. Then by the inductive hypothesis and the fact that $\omega(\iota_{n+1})\sigma$ does not introduce any new open minor branches, D contains the negation of the open literal on each minor branch that is necessary to derive the image $P(\bar{t}_P)$ for $P(\bar{t}_P)[D]\langle B\rangle$ in $\mathcal{F}_{n+1}$. Hence statement holds for $\mathcal{K}^{\mathsf{X}}_{n+1}$.

Case 2 ($R_1(\bar{x}_1) \wedge R_2(\bar{x}_2) \rightarrow P(\bar{y})$): Suppose inference ι_{n+1} takes $R_1(\bar{t}_{R_1})[D_1]\langle B_1\rangle$ and $R_2(\bar{t}_{R_2})[D_2]\langle B_2\rangle \in \mathcal{K}^{\mathsf{X}}_{n}$ and applies constraint $R_1(\bar{x}_1) \wedge R_2(\bar{x}_2) \rightarrow P(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P(\bar{t}_P)[D_1 \cup D_2]\langle B_1 \cup B_2\rangle$ in $\mathcal{K}^{\mathsf{X}}_{n+1}$. By Construction 4.17, $\mathcal{F}$ is obtained by attaching a copy of $\omega(\iota_{n+1})\sigma$ at the tip of each branch associated with branch number $B_1 \cup B_2$ which contains $R_1(\bar{t}_{R_1})[D_1]\langle B_1\rangle$ and $R_2(\bar{t}_{R_2})[D_2]\langle B_2\rangle$ on the branch. Analogous to case (1), the expansion at the tip of each branch extends the tip with open atom $P(\bar{t}_P)$, so $\omega(\iota_{n+1})\sigma$ does not introduce any new open minor branches. Then by the inductive hypothesis, since D_1, D_2 contain the negation of the open literal on each minor branch that is necessary to derive the images $R_1(\bar{t}_{R_1}), R_2(\bar{t}_{R_2})$ respectively, so does $D_1 \cup D_2$ for the image $P(\bar{t}_P)$. Hence, statement holds for $\mathcal{K}^{\mathsf{X}}_{n+1}$.

Case 3 ($R(\bar{x}) \rightarrow P_1(\bar{y}) \vee P_2(\bar{y})$): The proof is analogous to case (1) since the left hand side of constraint only contains a single atom. Suppose inference ι_{n+1} takes conditional atom $R(\bar{t}_R)[D]\langle B\rangle \in \mathcal{K}^{\mathsf{X}}_{n}$ and applies constraint $R(\bar{x}) \rightarrow P_1(\bar{y}) \vee P_2(\bar{y}) \in \mathsf{X}(\Sigma)$ to derive new conditional atom $P_1(\bar{t}_P)[D]\langle B \cup \{m : 0\}\rangle$ and $P_2(\bar{t}_P)[D]\langle B \cup \{m : 1\}\rangle$ in $\mathcal{K}^{\mathsf{X}}_{n+1}$, where m is a fresh branching point number. By Construction 4.17, the closing tableau $\mathcal{F}_{n+1}$ for $\mathcal{K}^{\mathsf{X}}_{n+1}$ is obtained by attaching a copy of $\omega(\iota_{n+1})\sigma$ at the tip of each branch associated with branch number B in $\mathcal{F}_n$. Then each expansion results in a new left closed branch with atomic closure $\neg R(\bar{v}_x), R(\bar{t}_R)$ under $\bar{v}_x\sigma = \bar{t}_R$ and a new right open disjunction of open atoms $P_1(\bar{t}_P)$ and $P_2(\bar{t}_P)$ at its leaf node. In other words, attaching $\omega(\iota_{n+1})\sigma$ at the tip of a branch extends that tip with open disjunction of $P_1(\bar{t}_P) \vee P_2(\bar{t}_P)$. Then by the inductive hypothesis and the fact that $\omega(\iota_{n+1})\sigma$ does not introduce any new open minor branches, D contains the negation of the open literal on each minor branch that is necessary to derive the image $P_1(\bar{t}_P)$ and $P_2(\bar{t}_P)$ for $P_1(\bar{t}_P)[D]\langle B \cup \{m : 0\}\rangle$ and $P_2(\bar{t}_P)[D]\langle B \cup \{m : 1\}\rangle$ in

$\mathcal{F}_{n+1}$ respectively. Hence statement holds for $\mathcal{K}^{\mathsf{X}}_{n+1}$.

Case 4 ($R(\bar{x}) \rightarrow \exists v.P_1(v, \bar{x})$): The proof is analogous to case (1) since the left hand side of constraint only contains a single atom.

Case 5 (physical rule): Let p denote a physical atom. Correctness follows from the fact that the expansion of $p \rightarrow p$ results in an open minor branch containing open literal $\neg p$ whose negation is in the dependencies of $p[p]\langle\rangle$. Since $p \rightarrow p$ is a tautology, (ι_{n+1}) is the minimal sequence that derives $p[p]\langle\rangle$, so $\{p\}$ indeed contains the negation of the open literal on the only minor branch that is necessary to derive the image p in $\mathcal{F}_{n+1}$. By the inductive hypothesis, the same is true for the remaining conditional atoms originally in $\mathcal{K}^{\mathsf{X}}_{n}$, so statement holds for $\mathcal{K}^{\mathsf{X}}_{n+1}$.

This covers all the cases, so by induction, statement holds. ■

Lemma 4.22 Let $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ be a $\mathfrak{K}$-tableau, and $\mathsf{CS}^{\mathsf{L}}, \mathsf{CS}^{\mathsf{R}}$ its left and right closing sets. Let $\mathcal{C}^{\mathsf{X}} = \{r_1[D_1]\langle B_1\rangle, \ldots, r_k[D_k]\langle B_k\rangle\}$ be the set of conditional atoms used in the construction of $s \in \mathsf{CS}^{\mathsf{X}}$. That is, $s = \{\neg r_1, \ldots, \neg r_k\} \cup D_1 \cup \cdots \cup D_k$. Let $(\iota_1, \ldots, \iota_n)$ be the minimal sequence of inferences that derives $\mathcal{C}^{\mathsf{X}}$ from the initial atoms $\mathcal{I}^{\mathsf{X}}$. Construct closing tableau $\mathcal{F}$ from $(\iota_1, \ldots, \iota_n)$ and $\mathcal{I}^{\mathsf{X}}$. Then the images for each $r_i[D_i]\langle B_i\rangle \in \mathcal{C}^{\mathsf{X}}$ are open leaf nodes in $\mathcal{F}$, and the remaining leaf nodes in $\mathcal{F}$ are on minor branches.

proof: By Construction 4.10, every B_i contains a branch number $m : d$ such that $m : d \notin B_j$ where $i \neq j$, so the images of distinct conditional atoms in $\mathcal{C}^{\mathsf{X}}$ appear on distinct branches in $\mathcal{F}$. Then the images for each $c \in \mathcal{C}^{\mathsf{X}}$ must be open leaf nodes in $\mathcal{F}$, or else $(\iota_1, \ldots, \iota_n)$ is not minimal since we can remove an inference and still derive all of $\mathcal{C}^{\mathsf{X}}$. Moreover, the leaf node of each major branch is an image of some $r_i[D_i]\langle B_i \in \mathcal{C}^{\mathsf{X}}$, or else either $(\iota_1, \ldots, \iota_n)$ is not minimal because we can remove a disjunctive inference rule and still derive $\mathcal{C}^{\mathsf{X}}$ or s is not a valid closing set because there exists m such that $\{m : 0, m : 1\} \nsubseteq B_1 \cup \cdots \cup B_k$. Then all remaining leaf nodes must be on minor branches. ■

Lemma 4.23 (Properties of Closing Tableau) Let $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ be a $\mathfrak{K}$-tableau, and $\mathsf{CS}^{\mathsf{L}}, \mathsf{CS}^{\mathsf{R}}$ its left and right closing sets. Let $\mathcal{C}^{\mathsf{X}} = r_1[D_1]\langle B_1\rangle, \ldots, r_k[D_k]\langle B_k\rangle$ be the set of conditional atoms used in the construction of $s \in \mathsf{CS}^{\mathsf{X}}$. Let $(\iota_1, \ldots, \iota_n)$ be the minimal sequence of inferences that derives $\mathcal{C}^{\mathsf{X}}$ from the initial atoms $\mathcal{I}^{\mathsf{X}}$. Construct closing tableau $\mathcal{F}$ from $(\iota_1, \ldots, \iota_n)$ and $\mathcal{I}^{\mathsf{X}}$. Then $\mathcal{F}$ satisfy the following:

1. $\mathcal{F}$ may be closed by attaching s to its root; and

2. The resulting closed tableau can only interpolate to $\bot(\top)$ for $\mathsf{X} = \mathsf{L}(\mathsf{X} = \mathsf{R})$ respectively.

proof: First, we prove property (2). By Construction 4.17, only constraints from $\mathsf{X}(\Sigma)$ are used during the construction of $\mathcal{F}$. By the construction of s, all literals attached to the root are implicitly X biased. Hence, the resulting closed tableau of $\mathcal{F}$ only contains X biased formulae, so by Definition 3.12, it can only interpolate to $\bot(\top)$ for $\mathsf{X} = \mathsf{L}(\mathsf{R})$ respectively. Hence, property (2) holds.

Next we prove property (1). Since $(\iota_1, \ldots, \iota_n)$ is the minimal sequence that derives $\mathcal{C}^{\mathsf{X}}$, by Lemma 4.22, the leaf node on each major branch in $\mathcal{F}$ is an image of some $c \in \mathcal{C}^{\mathsf{X}}$. Then attaching $\{\neg r_1, \ldots, \neg r_k\}$ to the root of $\mathcal{F}$ closes all major branches in $\mathcal{F}$. Next, observe that since $(\iota_1, \ldots, \iota_n)$ is minimal, by Lemma 4.21, each D_i contains the negation of the open literal on each minor branch on the major branch with leaf node that is an image of $r_i[D_i]\langle B_i\rangle$. Then attaching $D_1 \cup \cdots \cup D_k$ to the root of $\mathcal{F}$ closes all minor branches in $\mathcal{F}$. Combining the above, attaching $s = \{\neg r_1, \ldots, \neg r_k\} \cup D_1 \cup \cdots \cup D_k$ to the root of $\mathcal{F}$ closes $\mathcal{F}$. Thus, property (1) holds.

Thus, both properties hold for the closing tableau constructed from the minimal sequence of inferences $(\iota_1, \ldots, \iota_n)$ and initial atoms $\mathcal{I}^{\mathsf{X}}$. ■

Finally, we apply Construction 4.17 and Lemmas 4.16 and 4.23 to prove Theorem 4.13.

proof of Theorem 4.13: Let $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ and ψ be $\mathfrak{K}$-tableau and formula that satisfy the closing condition (Definition 4.12). Construct interpolating tableau $\mathcal{F}^{\psi}$. Since $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ and ψ satisfy the closing condition, we can construct left and right closing sets $\mathsf{CS}^{\mathsf{L}}, \mathsf{CS}^{\mathsf{R}}$ and left and right complementary closing sets $\mathsf{KS}^{\mathsf{L}}, \mathsf{KS}^{\mathsf{R}}$ such that for each $s' \in \mathsf{KS}^{\mathsf{X}}$, there exists $s \in \mathsf{CS}^{\mathsf{X}}$ such that $s \subseteq s'$. Construct closing tableau $\mathcal{F}_s^{\mathsf{X}}$ for s. Then, we can construct new $\mathfrak{F}$-tableau $\mathcal{F}$ by hanging a closing tableau $\mathcal{F}_s^{\mathsf{X}}$ at the tip of each open branch corresponding to s' in $\mathcal{F}^{\psi}$ for all pairs of s', s. By Lemma 4.23 property (1) and Lemma 4.16 property (1), $\mathcal{F}$ is a closed $\mathfrak{F}$-tableau. By Lemma 4.23 property (2) and Lemma 4.16 property (2), $\mathcal{F}$ interpolates to formula ψ' that is logically equivalent to ψ. Hence, ψ is a valid interpolant, as desired. ■

A notable consequence of Construction 4.10 is that if $r[D_1]\langle B_1\rangle$ and $r[D_2]\langle B_2\rangle$ are conditional atoms such that $D_2 \subseteq D_1$ and $B_1 \subseteq B_2$, then for each closing set s_2 constructed from $r[D_2]\langle B_2\rangle$, there exists a closing set s_1 constructed from $r[D_1]\langle B_1\rangle$ such that $s_1 \subseteq s_2$. Then, by Lemma 4.22, the derivation summarized by $r[D_2]\langle B_2\rangle$ is subsumed by the derivation summarized by $r[D_1]\langle B_1\rangle$. Hence, we can apply the following restriction to optimize theorem proving in $\mathfrak{K}$-tableau.

Definition 4.24 (Subsumption) *Let $r[D_1]\langle B_1\rangle$ and $r[D_2]\langle B_2\rangle$ be conditional atoms such that $D_2 \subseteq D_1$ and $B_1 \subseteq B_2$. We say $r[D_2]\langle B_2\rangle$ is subsumed by $r[D_1]\langle B_1\rangle$.*

Corollary 4.25 Let $\mathcal{F}^{\mathsf{X}}$ be a $\mathfrak{K}$-tableau and $r[D_1]\langle B_1\rangle$ and $r[D_2]\langle B_2\rangle$ conditional atoms in $\mathcal{F}^{\mathsf{X}}$. If $r[D_2]\langle B_2\rangle$ is subsumed by $r[D_1]\langle B_1\rangle$, then a conditional atom $r^*[D^*]\langle B^*\rangle$ is derivable from $\mathcal{F}^{\mathsf{X}}$ if and only if it is derivable from $\mathcal{F}^{\mathsf{X}}\backslash\{r[D_2]\langle B_2\rangle\}$ or subsumed by a conditional atom derivable from $\mathcal{F}^{\mathsf{X}}\backslash\{r[D_2]\langle B_2\rangle\}$.

proof: Follows from Lemma 4.22 and Definition 4.7. ■

Corollary 4.25 may be applied to remove all subsumed conditional atoms in a $\mathfrak{K}$-tableau.

4.3 Guessing Tautologies

In Section 4.1, we introduced the physical rule which inserts conditional atoms of the form $p[p]\langle\rangle$ for arbitrary ground physical atom p as a consequence of the tautology $\forall\bar{x}.(p \rightarrow p)$. However, the freedom to pick arbitrary p introduces significant non-determinism during theorem proving, which exacerbates the issue with the combinatorial explosion of the proof search space. Hence, physical rules must be applied with caution to avoid introducing extraneous conditional atoms. We illustrate the gravity of this issue with an example.

Example 4.26 Define input schema $\Sigma = \{R(x) \leftrightarrow P(x), O_i(x) \leftrightarrow O_{i+1}(x) \mid \forall i \in \mathbb{N}, i \leq 1000\}$, access paths $\mathsf{S_A} = \{P/1/0, O_i/1/0 \mid \forall i \in \mathbb{N}, i \leq 1001\}$, and user query $q(x) \leftrightarrow R(\bar{t}_R)$.

It is clear from the example that $O_i(\bar{t}_{O_i})$ is extraneous for all i, so applying physical rules indiscriminately results in 1000 extraneous inferences and closing sets. Moreover, the problem illustrated in this simple example can be much worse in reality for more complicated schemata containing existential variables and disjunctions.

4.3.1 Basic Physical Rule [25]

Fortunately, it is possible to apply physical rules as selectively as necessary. Two simple observations are sufficient for eliminating most extraneous physical atoms.

Observation 4.27 By Lemma 4.16, if an atom r appears in an interpolant ψ, then there exists left and right complementary closing sets $\mathsf{KS^L}$ and $\mathsf{KS^R}$ for ψ, each containing either

r or $\neg r$, although r and $\neg r$ may not be necessary to closed the interpolating tableau for ψ. In other words, r appears in either both KS^{L} and KS^{R} or neither.

We do not know in advance which literals are necessary to closed the interpolating tableau for ψ, but we do know that if one of r and $\neg r$ is possibly needed on the X side, then by Observation 4.27, the same can be said about the $\overline{\mathsf{X}}$ side. It is thus constructive to restrict the application of physical rules by the following condition.

Definition 4.28 (Basic Physical Rule) *If $P(\bar{t})[D]\langle B\rangle$ is in $\mathcal{K}^{\mathsf{X}}$ where P is physical, then insert $P(\bar{t})[P(\bar{t})]\langle\rangle$ in $\mathcal{K}^{\overline{\mathsf{X}}}$ if not already exists.*

The above condition is sufficient if we only wish to enumerate query plans (i.e. interpolants) consisting of literals whose atoms are consequences of the schema Σ and user query q. Otherwise it is clear that the condition in Definition 4.28 can never be met. In most cases, if an atom r is not a consequence of Σ and q, r appears in a tautology in the interpolant ψ, otherwise $\Sigma \cup \{q\} \to r$ which is a contradiction.

4.3.2 A Messy Guessing Game

The question remains whether it is constructive to apply physical rules for literals whose atoms are not consequences of the schema and user query. The following example illustrates a scenario in which applying physical rule for such an atom produces an optimal plan that is otherwise not found.

Example 4.29 (Negative Literal) Consider a query optimization problem for the user query $a(x) \wedge c(x)$ with respect to the database schema

Schema Constraints	Query		Normalized Constraints
	$q(x) \leftrightarrow a(x) \wedge c(x)$	$\xrightarrow{\text{trans}}$	$q(x) \to a(x)$ $q(x) \to c(x)$ $a(x) \wedge c(x) \to q(x)$
$c(x) \wedge b(x,y) \to \bot$		$\xrightarrow{\text{trans}}$	$c(x) \wedge b(x,y) \to \bot$

presented as a set of constraints and the set of assess paths $\mathsf{S_A} = \{a/1/0, b/2/0, c/1/0\}$. The system will still generate the query plan $a(x) \wedge c(x)$, but not the plan

$$a(x) \wedge (\neg\exists y.b(x,y)) \wedge c(x),$$

even though the schema *implies* $c(x) \rightarrow \neg\exists y.b(x,y)$. The second plan may be more efficient than the first plan when a is large, when checking the satisfaction of $\neg\exists y.b(x,y)$ is relatively inexpensive compared to checking the satisfaction of $c(x)$, and when the intersection of x values in $a(x)$ and $b(x,y)$ is now close to the size of a.

The failure to derive the second plan in this case can be traced to the forward chaining design of $\mathfrak{K}$-tableau: the premise $c(x) \wedge b(x,y) \rightarrow \bot$ is never executed since atoms for b cannot be derived by forward chaining.

We observe from the above example that $\neg r$ can be a consequence of Σ and q when r is not. Moreover, it is possible that an optimal plan contains $\neg r$ non-trivially (i.e., not in a tautology). Hence, it is constructive to apply the physical rule for r under the condition that its negation is a consequence of Σ and q. Unfortunately, since $\mathfrak{K}$-tableau performs forward chaining to derive positive ground literals only, r cannot be derived using just the basic physical rule alone. A straightforward way to rectify this problem is to extend the tableau construction by allowing negated atoms alongside the positive atoms. This, however, leads immediately to the need for handling free/universally quantified variables, unification, etc. Instead, we present an alternative solution inspired by the *magic set transformation* (MST) [5] that allows $\mathfrak{K}$-tableau construction to be based on efficient forward chaining of constraints over ground atoms while providing a general solution to the problem illustrated by Example 4.29.

A Messy Guessing Game. Before we present the MST-based transformation procedure, let us first consider the intuition our procedure aims to capture: guessing and generating tautologies $P(\bar{t})[P(\bar{t})]\langle\rangle$ with appropriate arguments $\bar{t}$ for the missing physical atoms. Let us illustrate this guessing game with Example 4.29.

Example 4.29 (Continued) Recall that $c(x) \wedge b(x,y) \rightarrow \bot$ is never executed because none of the constraints contains $b(x,y)$ on the right hand side. We initialize the $\mathfrak{K}$-tableau $\mathcal{K} = (\mathcal{K}^{\mathsf{L}}, \mathcal{K}^{\mathsf{R}})$ with $q(0)$ in $\mathcal{K}^{\mathsf{L}}$. Applying forward chaining, the left closing sets for Example 4.29 are $\{\neg a(0)\}$ and $\{\neg c(0)\}$, and the right closing set is $\{a(0), c(0)\}$. However, to generate the query plan '$a(x) \wedge (\neg\exists y.b(x,y)) \wedge c(x)$', we are missing the left closing set $\{b(0,1)\}$ because the system fails to generate its corresponding conditional atom $\bot[b(0, f^{\mathsf{R}}())]$, where $f^{\mathsf{R}}()$ is a R biased Skolem term for b. To introduce b in the tableau, we mirror the basic physical rule and insert tautologies $b(t_1,t_2)[b(t_1,t_2)]$, where t_1, t_2 are biased Skolem terms or initial Skolem constants. The task remains to *guess appropriate Skolem constants* for t_1, t_2 so that $\bot[b(0, f^{\mathsf{R}}())]$ is generated. In this simple case, we obviously guess $t_1 = 0$ and $t_2 = f^{\mathsf{R}}()$. However, in most cases, guessing appropriate arguments is a non-trivial task.

Example 4.29 shows how a simple guessing game can generate appropriate conditional

atoms for missing plans. In the next example, we illustrate difficulties with chaining of negative atoms, which can make the guessing game messy. We also see how MST is used to solve the messy guessing game.

Example 4.30 Consider a database schema $\Sigma = \{q(x) \rightarrow d(x), q(x) \rightarrow a(x), d(x) \wedge a(x) \rightarrow q(x), a(x) \wedge b(x,y) \rightarrow \exists z.c(x,y,z), b_1(x,z) \wedge b_2(z,y) \rightarrow b(x,y)\}$ for a user query q and a set of physical atoms $\{a, b_1, b_2, c, d\}$.

This example contains both existential variables and chaining of negative atoms $b(x,y)$, $b_1(x,z)$, $b_2(z,y)$. The current system fails to produce conditional atoms for physical predicates b_1, b_2, c, resulting in missing plan '$d(x) \wedge \neg\exists y, z, u.((b_1(x,z) \wedge b_2(z,y)) \wedge \neg c(x,y,u)) \wedge a(x)$'.

To produce appropriate argument bindings in plans, we must make educated guesses for $c(x,y,u)$ depending on previous guesses for $b(x,y)$. Similarly, we guess $b(x,y)$ depending on previous guesses for $b_1(x,z)$ and $b_2(z,y)$. Hence, in the presence of chaining, the guessing game becomes messy since previously guessed arguments must be back propagated into new guesses. The key observation is that any solution that plays the messy guessing game correctly must perform back chaining on previously guessed arguments.

Since MST is a rewriting procedure designed to simulate back chaining using forward chaining evaluation, it precisely solves the messy guessing game for our forward chaining system. Ultimately, the goal is to guess as few conditional atoms as needed to extend the query plan space. Given the above considerations, MST-based rewriting provides a well-rounded solution given the positive and forward chaining nature of the system.

4.3.3 Magic Physical Rule

Magic Set Transformation for $\mathfrak{K}$-Tableau [18]. As illustrated by Example 4.30, an MST-like rewriting procedure is necessary for argument back propagation. The idea is to create auxiliary magic atoms which serve as medium for back chaining needed to communicate argument bindings.

The rewriting procedure transforms a set of ANF input constraints Σ^o coupled with a set of access paths $\mathsf{S_A}$ into MST-enhanced constraints $\Sigma^o \cup \Sigma^g \cup \Sigma^p$, where Σ^g are the constraints that generate magic atoms and Σ^p the physical rules that generate missing conditional atoms from magic atoms. The procedure consists of 3 stages: (a) a first stage that initializes the set of magic atoms V^m from constraints with conjunction on their left hand sides; (b) a second stage that recursively computes magic atoms from right hand sides and uses V^s to record explored magic atoms; and (c) a final stage that computes the set of magic physical rules Σ^p.

1. Initial Transformation. For each constraints in Σ^o of the form

$$a(\overline{x}_a) \wedge b(\overline{x}_b) \rightarrow c(\overline{x}_c),$$

compute variable overlap $\overline{x}_{ab} = \overline{x}_a \cap \overline{x}_b$, and adornment string $A \in \{0,1\}^l$. Here, l is the number of arguments in $\overline{x}_b$, and $A_i = 1$ if the i^{th} argument in $\overline{x}_b$ appears in $\overline{x}_{ab}$ ($A_i = 0$ otherwise). Create predicate $magic\_b^A(\overline{x}_{ab})$ for predicate b with respect to argument binding $\overline{x}_{ab}$. Add new constraint

$$a(\overline{x}_a) \rightarrow magic\_b^A(\overline{x}_{ab})$$

to Σ^g, and add new predicate $magic\_b^A$ to V^m. Note that conditional atoms $magic\_b^A(\overline{x}_{ab})[]\{\}$ always have empty conditions and branches. Repeat for constraint $b(\overline{x}_b) \wedge a(\overline{x}_a) \rightarrow c(\overline{x}_c)$ where arguments are passed from b to a.

2. Chain Transformation. While V^m is non-empty, select $magic\_a^A \in V^m$, remove it from V^m, and add it to V^s. For each binary constraint

$$b(\overline{x}_b) \wedge c(\overline{x}_c) \rightarrow a(\overline{x}_a)$$

with the right hand side matching $magic\_a^A$, compute the subset of arguments $\overline{x}_a^A$ from $\overline{x}_a$ indicated by A, and add new constraint

$$magic\_a^A(\overline{x}_a^A) \rightarrow magic\_b^{A'}(\overline{x}_{ab}^A)$$

to Σ^g, where $\overline{x}_{ab}^A = \overline{x}_a^A \cup \overline{x}_b$. Then, add one additional new constraint

$$magic\_a^A(\overline{x}_a^A) \wedge b(\overline{x}_b) \rightarrow magic\_c^{A'}(\overline{x}_{abc}^A)$$

to Σ^g, where arguments $\overline{x}_{abc}^A = (\overline{x}_a^A \cup \overline{x}_b) \cap \overline{x}_c$, and new adornment string A' computed from $\overline{x}_{abc}^A$ and $\overline{x}_c$. Repeat for the constraint $c(\overline{x}_c) \wedge b(\overline{x}_b) \rightarrow a(\overline{x}_a)$ where arguments are passed from c to b. Transformations for all remaining constraints are defined analogously. Add new predicates to V^m if not already in $V^m \cup V^s$.

3. Magic Physical Rule. For each predicate $magic\_a^A$ corresponding to some physical predicate a, add a magic physical rule (symmetric to the basic physical rule) to Σ^p such that $magic\_a^A(\overline{x})$ generates conditional atom $a(\overline{x}_a)[a(\overline{x}_a)]$ on the same side of the tableau for its corresponding non-magic predicate, where the arguments indicated by A are copied from $\overline{x}$, and the rest of the arguments are fresh Skolem terms with the opposite bias.

Soundness and Efficacy. Previous work on interpolation-based query optimizers show that $\mathfrak{R}$-tableau are sound even when restricted by absorption. Now we show that the MST-enhanced constraints are sound and effectively generate appropriate negative conditions without using negative ground literals.

Theorem 4.31 (Soundness of MST-enhanced Constraints [18]) Given original input constraints Σ^o in ANF, the MST-enhanced rewriting of Σ^o, denoted by Σ^m, is sound.

proof: Follows from the correctness of MST and the fact that magic physical rules only insert tautologies, which preserve soundness. Hence, since Σ^o is sound, Σ^m is sound. ■

In addition to soundness, the patterns seen in Examples 4.29 and 4.30 generalize easily to complex input schemata containing arbitrarily many negative implications and long chains for which the missing physical atoms cannot be identified by inspection.

Theorem 4.32 (Efficacy of MST-enhanced Constraints [18]) Let Σ^o be the set of original input constraints in ANF and Σ^m be its MST-enhanced rewriting. If a negative atom is logically implied on some branch of the conditional tableau, then it is generated with the correct dependencies and branches by forward chaining with respect to Σ^m.

proof: The base case is identical to Example 4.29. Efficacy generalizes to schemata with chaining by induction on the length of chains. ■

Example 4.29 (revisited) We now illustrate the MST-based rewriting procedure with Example 4.29. Performing the MST-based rewriting procedure on ANF input constraints results in the following MST-enhanced constraints (some constraints are redundant, but included for the sake of completeness):

$$
\begin{aligned}
\Sigma^o =& \{q(x) \rightarrow a(x), q(x) \rightarrow c(x), a(x) \wedge c(x) \rightarrow q(x), c(x) \wedge b(x,y) \rightarrow \bot\};\\
\Sigma^g =& \{c(x) \rightarrow magic\_b^{10}(x), b(x,y) \rightarrow magic\_c^{1}(x), a(x) \rightarrow magic\_c^{1}(x),\\
& \quad c(x) \rightarrow magic\_a^{1}(x), magic\_a^{1}(x) \rightarrow magic\_q^{1}(x), magic\_c^{1}(x) \rightarrow magic\_q^{1}(x),\\
& \quad magic\_q^{1}(x) \rightarrow magic\_a^{1}(x), magic\_q^{1}(x) \wedge a(x) \rightarrow magic\_c^{1}(x)\}; \text{ and}\\
\Sigma^p =& \{magic\_b^{10}(x) \rightarrow \exists y.b(x,y), magic\_c^{1}(x) \rightarrow c(x), magic\_a^{1}(x) \rightarrow a(x)\}
\end{aligned}
$$

Executing the above MST-enhanced constraints using forward chaining produces the following $\mathfrak{K}$-tableau, where the missing atom $\bot[b(0, f^{\mathsf{R}}())]$ is generated.

$\mathcal{K}^L$		$\mathcal{K}^R$
$q(0)[]\langle\rangle$		
$a(0)[]\langle\rangle$	$\xrightarrow{\text{phys}}$	$a(0)[a(0)]\langle\rangle$
$c(0)[]\langle\rangle$	$\xrightarrow{\text{phys}}$	$c(0)[c(0)]\langle\rangle$
		$q(0)[a(0), c(0)]\langle\rangle$
		$\bot[a(0), c(0)]\langle\rangle$
$magic\_b^{10}(0)$		
$b(0, f^{\mathsf{R}}())[b(0, f^{\mathsf{R}}())]\langle\rangle$		
$\bot[b(0, f^{\mathsf{R}}())]\langle\rangle$		

Hence, the MST rewriting procedure effectively generates the missing physical atoms without additional reasoning beyond rule transformation. Ultimately, this yields the sought-after plan $a(x) \wedge (\neg\exists y.b(x, y)) \wedge c(x)$. Similarly, applying the MST-based rule rewriting procedure to Example 4.30 yields MST-enhanced rules that generate the missing query plans even in the presence of chaining.

Chapter 5

System Implementation

In the previous chapter, we define $\mathfrak{K}$-tableau and the closing condition, which can be used to recognize valid interpolants. To simplify the argument, we assume that candidate formulae are supplied by a black box. However, in reality, being able to enumerate promising candidates efficiently is equally important and should not be overlooked. In this chapter, we present a solution, based on the A^* search algorithm, that addresses the problem of efficient candidate enumeration. Moreover, we summarize the system design of an existing interpolation-based query optimizer which leverages $\mathfrak{K}$-tableau and A^* search for plan enumeration.

5.1 Enumerating Plans

A key observation about plan enumeration is that we are not completely clueless about the features of promising candidates. In fact, the closing sets already provide much guidance for generating promising candidates. The idea is to iteratively construct more complex candidate formulae from simpler formulae called *fragments* based on guidance from the closing sets. However, note that depending on the $\mathfrak{K}$-tableau constructed, the set of interpolants that are recognized by the closing sets may be a subset of the set of all valid interpolants. In the context of $\mathfrak{K}$-tableau interpolation, we assume that interpolants refer to interpolants that are recognized by a specific set of closing sets.

Definition 5.1 (Fragment) *A fragment is a subformula of a candidate formula, or alternatively, fragments are formulae that can be combined using logical symbols to construct candidate formulae. We call a fragment* useful *if it appears non-trivially (i.e., not in a*

tautology) in an interpolant that is recognized by the closing sets. If a fragment is not useful then we say it is extraneous.

In some sense, fragments and candidates are essentially the same. Every fragment ρ is initially generated as a candidate; if ρ fails the closing condition, it is either pruned out for being extraneous or kept as a fragment for constructing future candidates. From hereon we use candidate and fragment interchangeably. We initialize the set of fragments by literals, which are the most basic constituents of interpolants. Observe that by the closing condition, a literal is only useful if it appears in the closing sets. Then we can further restrict the set of initial fragments by only picking literals that appear in the closing sets.

Intuitively, we can think of the plan enumeration problem as a jigsaw puzzle. The closing sets serve as a guiding picture for which we aim to reconstruct with fragments which serve as jigsaw pieces in the plan enumeration puzzle. Each time we construct a bigger fragment, we can compare it against the closing sets to measure its progress towards the guiding picture.

To iteratively construct increasingly complex candidates, we leverage A^* search which uses a heuristic function based on closing sets to explore the more promising fragments before the less promising ones. The heuristic function uses the set differences between closing sets and complementary closing sets for ρ to estimate the distance between ρ and a valid interpolant. At each step, we can prune out extraneous fragments by requiring the intersection between the closing sets and each complementary closing set for the fragment is non-empty. Moreover, the cost model for query plans is built into the heuristic function, so A^* search always explores fragments that are estimated to lead to cheaper query plans first.

Finally, to ensure that all output interpolants are valid query plans, for each fragment, we check whether all binding patterns are satisfied and prune out those with invalid binding patterns.

5.2 The Interpolation Test Bed

The interpolation test bed (ITB) is an existing interpolation-based query optimizer that implements $\mathfrak{K}$-tableau and A^* search for query plan enumeration. The system design is summarized in Figure 5.1.

A notable difference between $\mathfrak{K}$-tableau and its implementation in ITB is that biased Skolem terms are replaced by Skolem constants, each associating with an age number

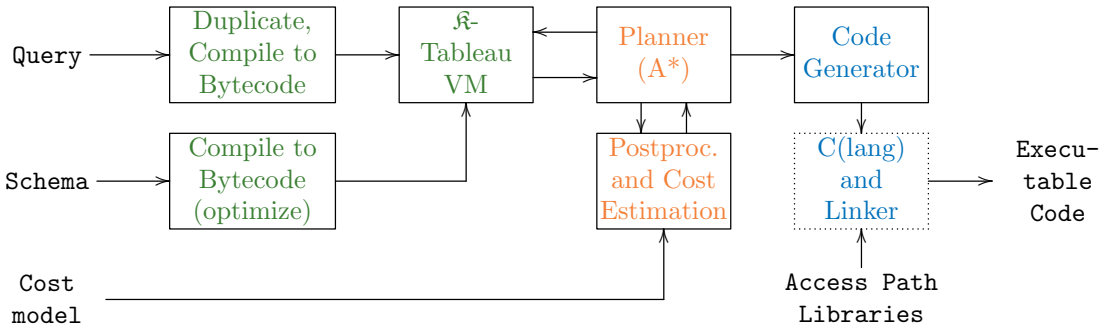


Figure 5.1: ITB architecture [32].

denoting the order of creation. Since the order or creation is a linear extension of the partial order on biased Skolem terms (i.e., the subterm property), we can correctly quantify Skolem constants in the order of the most to the least recently created.

The modules that are marked in green correspond to a programmed virtual machine that implements a $\mathfrak{K}$-tableau. Notice that user queries and constraints in ANF are compiled into bytecodes which program the virtual machine. It is clear that the inference procedure can be implemented using a straightforward approach By Proposition 4.2: matching the left hand side of each constraint with conditional atoms in the tableau. However, due to the combinatorial explosion of the proof search space (i.e., the set of all conditional atoms in the tableau), the search for conditional atoms can be costly when implementing the straightforward approach. Alternatively, we can implement the inference procedure by selecting a new conditional atoms and executing all constraints with matching left hand sides, which avoids the issue with search and simplifies the implementation since the set of matching constraints is fixed for each predicate symbol. The following example illustrates how constraints are implemented by bytecodes.

Example 5.2 (Constraints to Bytecodes) Constraints are translated into bytecodes as follows:

1. $R(x) \to \exists v.R_2(v, x)$ is translated into bytecode '`MKSKOLEM` R_2' for predicate R, which makes new conditional atoms for $R_2(c, x)$ with fresh Skolem constant c, argument x, and appropriate dependencies and branches.

2. $R_3(x) \to R_4(x) \vee R_5(x)$ is translated into bytecode '`MKOR` R_4 x 0 `MKOR` R_5 x 1' for predicate R_3, which makes new conditional atoms for $R_4(x)$ and $R_5(x)$ at a fresh branching point with direction numbers 0 and 1 respectively.

3. $R(x) \wedge R_6(x, y) \rightarrow R_7(x)$ is translated into bytecode '`JOIN` R_6 `ON` x, y `MKATOM` R_7 x' for predicate R, which performs join on conditional atoms for $R_6(x, y)$ and makes new conditional atoms for $R_7(x)$.

In addition, for physical predicates $P \in \mathsf{S_A}$, we also include the bytecode

4. '`MKPHYS`', which executes the basic physical rule by inserting tautological conditional atoms for P into the complementary tableau if not already exist.

The modules that are marked in orange correspond to plan enumeration based on A^* search. Note that ITB may alternate between theorem proving and query planning in attempt to improve closing sets and extend the space of recognized interpolants.

Finally, the modules marked in blue correspond to program generation for the query plan. The program generation process is briefly described in Section 2.1.3. The output of the ITB is an executable program that retrieves answers for the user query. Detailed description and report on ITB's system design can be found in [32].

Chapter 6

Beyond This Thesis

6.1 Related Work

Query optimization has been studied extensively since the proposal of the relational model by Codd. In the context of this thesis, we only consider query optimization via reformulation under constraints. The two most prominent approaches to *query reformulation under constraints* are Chase & Backchase (CB) [15] and Craig interpolation [9, 11, 25, 31]. It has been noted in [31] that CB may be thought of as a special case of Craig interpolation. Detailed reduction and analysis can be found in [9, 31].

Other than $\mathfrak{F}$-tableau and $\mathfrak{K}$-tableau, many variant tableaux systems support interpolation for first-order logic [21, 19, 36]. In addition, interpolation methods have also been developed for non-tableau-based systems such as resolution provers [24].

6.2 Future Directions

1. In Section 4.1.1, we remark that all nested conjunctions on the left hand sides of implications can be transformed into binary conjunction by inserting auxiliary atoms. However, it is remains open whether **binary conjunction** will outperform nested conjunctions for the purpose of theorem proving since it is not clear whether the overhead of creating auxiliary atoms outweighs the cost of nested conjunctions.

2. In the definition of $\mathfrak{F}$-tableau and $\mathfrak{K}$-tableau, we assume all disjunctions are binary although this does not have to be the case. It is easy to modify the inference and

interpolation rules to accommodate for **multi-way disjunctions**. Nonetheless, it remains open whether one is better than the other performance-wise.

3. The naïve approach to implementing **equality** is to apply the standard equality axioms [19], which is what we assume for $\mathfrak{F}$-tableau and $\mathfrak{K}$-tableau in this thesis. Unfortunately, this naïve approach is known to lead to combinatorial explosion of the proof search space. To handle equalities more efficiently, special machineries has been applied to proof systems. For example, superposition calculus is developed as a result of applying paramodulation to resolution. It would be interesting and useful to know whether similar techniques can generalize to interpolation.

4. Craig interpolation is not limited to tableau proof systems and is supported by **alternative proof systems** such as resolution provers. It remains open whether non-tableau provers, such as resolution and superposition provers, can outperform tableau-based systems with respect to interpolation for the purpose of query optimization.

References

[1] Serge Abiteboul and Oliver M. Duschka. Complexity of answering queries using materialized views. In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '98, page 254–263. Association for Computing Machinery, 1998.

[2] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] Charles W. Bachman. Summary of current work ansi/x3/sparc/study group: Database systems. *SIGMOD Rec.*, 6(3):16–39, jul 1974.

[4] Charles W. Bachman, Richard E. Batchelor, I. Marvin Beriss, Charles R. Blose, Turgut I. Burakreis, Vincent Delia Valle, George G. Dodd, William Helgeson, John Lyon, A. (Tax) Metaxides, Gerald E. McKinzie, Paul Siegel, Warren G. Simmons, Larry L. Sturgess, Harrison Tellier, Sharon B. Weinberg, and George T. Werner. *Data Base Task Group Report to the CODASYL Programming Language Committee, October 1969*. Association for Computing Machinery, New York, NY, USA, 1969.

[5] François Bancilhon, David Maier, Yehoshua Sagiv, and Jeffrey D. Ullman. Magic sets and other strange ways to implement logic programs (extended abstract). In *PODS '86*, 1985.

[6] C. Beeri and R. Ramakrishnan. On the power of magic. In *Proceedings of the Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '87, page 269–284, New York, NY, USA, 1987. Association for Computing Machinery.

[7] Michael Benedikt. How can reasoners simplify database querying (and why haven't they done it yet)? In *ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, SIGMOD/PODS '18, page 1–15, 2018.

[8] Michael Benedikt, Julien Leblay, and Efthymia Tsamoura. Pdq: Proof-driven query answering over web-based data. *Proc. VLDB Endow.*, 7(13):1553–1556, aug 2014.

[9] Michael Benedikt, Balder Ten Cate, and Efthymia Tsamoura. Generating plans from proofs. *ACM Trans. Database Syst.*, 40(4), feb 2016.

[10] E. W. Beth. On padoa's method in the theory of definition. *Journal of Symbolic Logic*, 21(2):194–195, 1956.

[11] 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 Sonia Bergamaschi, Stefano Lodi, Riccardo Martoglia, and Claudio Sartori, editors, *Proceedings of the Eighteenth Italian Symposium on Advanced Database Systems, SEBD 2010, Rimini, Italy, June 20-23, 2010*, pages 426–437. Esculapio Editore, 2010.

[12] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970.

[13] E. F. Codd. Relational completeness of data base sublanguages. *Research Report / RJ / IBM / San Jose, California*, RJ987, 1972.

[14] William Craig. Three uses of the Herbrand-Gentzen theorem in relating model theory and proof theory. *The Journal of Symbolic Logic*, 22(3):269–285, 1957.

[15] Alin Deutsch, Lucian Popa, and Val Tannen. Physical data independence, constraints, and optimization with universal plans. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, page 459–470, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[16] Ramez Elmasri and Shamkant B. Navathe. *Fundamentals of Database Systems, 7th Edition*. Benjamin/Cummings, 2015.

[17] Herbert B. Enderton. *A mathematical introduction to logic*. Academic Press, 1972.

[18] Eva Feng, David Toman, and Grant E. Weddell. Magic sets in interpolation-based rule driven query optimization. In Guido Governatori and Anni-Yasmin Turhan, editors, *Rules and Reasoning - 6th International Joint Conference on Rules and Reasoning, RuleML+RR 2022, Berlin, Germany, September 26-28, 2022, Proceedings*, volume 13752 of *Lecture Notes in Computer Science*, pages 198–207. Springer, 2022.

[19] Melvin Fitting. *First-Order Logic and Automated Theorem Proving, Second Edition.* Graduate Texts in Computer Science. Springer, 1996.

[20] G. Graefe. Volcano— an extensible and parallel query evaluation system. *IEEE Trans. on Knowl. and Data Eng.*, 6(1):120–135, feb 1994.

[21] Reiner Hähnle and Peter H. Schmitt. The liberalized delta-rule in free variable semantic tableaux. *J. Autom. Reason.*, 13(2):211–221, 1994.

[22] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[23] Ian Horrocks and Stephan Tobies. Optimisation of terminological reasoning. In Franz Baader and Ulrike Sattler, editors, *Proceedings of the 2000 International Workshop on Description Logics (DL2000), Aachen, Germany, August 17-19, 2000*, volume 33 of *CEUR Workshop Proceedings*, pages 183–192. CEUR-WS.org, 2000.

[24] Guoxiang Huang. Constructing craig interpolation formulas. In *COCOON*, volume 95, pages 181–190, 1995.

[25] Alexander K. Hudek, David Toman, and Grant E. Weddell. On enumerating query plans using analytic tableau. In Hans de Nivelle, editor, *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015*, volume 9323 of *Lecture Notes in Computer Science*, pages 339–354, 2015.

[26] Laura Kovács and Andrei Voronkov. First-order theorem proving and vampire. In Natasha Sharygina and Helmut Veith, editors, *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*, volume 8044 of *Lecture Notes in Computer Science*, pages 1–35. Springer, 2013.

[27] Barbara Liskov and Stephen Zilles. Programming with abstract data types. In *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*, page 50–59, New York, NY, USA, 1974. Association for Computing Machinery.

[28] Raghu Ramakrishnan. *Database Management Systems, 3rd Ed.* WCB/McGraw-Hill, 2002.

[29] 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*, AAAI'14, page 2344–2351. AAAI Press, 2014.

[30] Stephan Schulz, Simon Cruanes, and Petar Vukmirović. Faster, higher, stronger: E 2.3. In Pascal Fontaine, editor, *Proc. of the 27th CADE, Natal, Brasil*, number 11716 in LNAI, pages 495–507. Springer, 2019.

[31] David Toman and Grant Weddell. *Fundamentals of Physical Design and Query Compilation*. Morgan & Claypool Publishers, 2011.

[32] David Toman and Grant E. Weddell. An interpolation-based compiler and optimizer for relational queries (system design report). In Thomas Eiter, David Sands, Geoff Sutcliffe, and Andrei Voronkov, editors, *IWIL@LPAR 2017 Workshop and LPAR-21 Short Presentations*, volume 1 of *Kalpa Publications in Computing*. EasyChair, 2017.

[33] David Toman and Grant E. Weddell. Using feature-based description logics to avoid duplicate elimination in object-relational query languages. *Künstliche Intell.*, 34(3):355–363, 2020.

[34] David Toman and Grant E. Weddell. FO rewritability for OMQ using beth definability and interpolation. In Martin Homola, Vladislav Ryzhikov, and Renate A. Schmidt, editors, *Proceedings of the 34th International Workshop on Description Logics (DL 2021) part of Bratislava Knowledge September (BAKS 2021), Bratislava, Slovakia, September 19th to 22nd, 2021*, volume 2954 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2021.

[35] David Toman and Grant E. Wedell. Projective Beth Definability and Craig Interpolation for relational query optimization (material to accompany invited talk). In Renate A. Schmidt, Christoph Wernhard, and Yizheng Zhao, editors, *Proceedings of the Second Workshop on Second-Order Quantifier Elimination and Related Topics (SOQE 2021)*, volume 3009 of *CEUR Workshop Proceedings*, pages 1–13. CEUR-WS.org, 2021.

[36] Christoph Wernhard. Craig interpolation with clausal first-order tableaux. *J. Autom. Reason.*, 65(5):647–690, 2021.