Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Automated Reasoning to Infer all Minimal Keys*

P. Cordero, M. Enciso, A. Mora
Universidad de Málaga, Spain
pcordero@uma.es, enciso@lcc.uma.es, amora@ctima.uma.es

Abstract

Wastl introduced for first time a tableaux-like method based on an inference system for deriving all minimal keys from a relational schema. He introduced two inference rules and built an automated method over them. In this work we tackle the key finding problem with a tableaux method, but we will use two inference rules inspired by the Simplification Logic for Functional Dependencies. Wastl's method requires the input to be a set of functional dependencies (FDs) with atomic right hand sides. Therefore, it is necessary to apply *fragmentation rule* with the consequent increasing of the input. The main novelty of our rules is that they deal with generalized formulas, avoiding the fragmentation needed in the former tableaux. Finally we illustrate the advantages of our new tableaux method with an experiment.

1 Introduction

The notion of a key is fundamental to any data model, including Codd's relational model of data [2]. Besides primary keys, unique constraints (candidate keys) provide a more complete understanding of the model.

Keys constraints specify the sets of attributes of a relation such that their projection univocally identifies each tuple of the relation. Some basic concepts concerning keys are summarized in the following.

Let $\mathcal{A}$ be a non-empty and finite set of attributes. A relation is a set of tuples $R \subseteq 2^{\mathcal{A}}$. An expresion $X \rightarrow Y$ with $X, Y \subseteq \mathcal{A}$ is named functional dependency (FD). A relation R satisfies $X \rightarrow Y$ if, for all pair of tuples $t_1, t_2 \in R$ if they agree on X, they agree on Y.

The notion of functional dependency allows to characterize the keys for a given relation as those non-empty subset $X \subseteq \mathcal{A}$ such as R satisfies $X \rightarrow \mathcal{A}$.

Keys are not only fundamental to data design, they are also considered powerful tools to solve several problems related to data query and management. In [9] the authors affirm that *'identification of keys is a crucially important task in many areas of modern data management, including data modeling, query optimization (provide a query optimizer with new access paths that can lead to substantial speedups in query processing), indexing (allow the database administrator to improve the efficiency of data access via physical design techniques such as data partitioning or the creation of indexes and materialized views), anomaly detection, and data integration'*.

A very outstanding characteristic of keys is their minimality. Thus, in the literature, the key finding problem is focussed on minimal keys, i.e. keys with no superfluous attributes.

The extraction of minimal keys for a given relation where a set of FDs hold has been well studied. As a preliminary work, keys were studied within the framework of the implication matrix in [4]. But the algorithm of Lucchesi and Osborn [5] to find all the keys in a relational scheme is considered the first deep study around this problem and it is the most cited work up until now. Their work also provides some interesting results regarding the complexity of the problem: Osborn shows in her Ph.D. thesis that *'the number of minimal keys for a relational system can be exponential in the number of attributes or factorial in the number of dependencies and that both of these upper bounds are attainable'*. Furthermore, Yu and Johnson [11] established that the number of keys is limited by the factorial of the number of dependencies, so, there does not exists a polynomial algorithm for this problem. Other likely intractability results are also well-known, e.g. it is NP-complete to decide whether there is a key of size at most k, given a set of FDs [5].

All the classical algorithms use the closure paradigm to check if a given subset of attributes is a key with regard to a set of FDs. Some authors have used several techniques to tackle the key finding problem. Saiedian and Spencer [8] propose an algorithm for computing the candidate keys using attribute graphs, but it can be only applied when the graph of FDs is not strongly connected. Recently, Zhang in [12] use Karnaugh maps to calculate all the keys.

In this work, we follow the approach of R. Wastl who introduces in [10] for first time a Hilbert style inference system, named $\mathbb{K}$, for deriving all keys of a relation schema. He also affirms that *"the method developed for computing keys is useful in algorithms of 3NF-normalization of relational*

*Supported by grants No. P09-FQM-5233 of the Junta de Andalucía and No. TIN2011-28084 of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF).

database'. R. Wastl builds a tableaux which represents the search space to find all the keys. Each step in the tableaux construction is guided by the application of the inference system $\mathbb{K}$.

In this paper an improvement of Wastl's algorithm is presented. We have developed a novel Hilbert-style algorithm strongly based on the Simplification Logic for Functional Dependencies (namely $\mathbf{SL}_{FD}$) [6]. The inference system of $\mathbf{SL}_{FD}$ is equivalent to Armstrong's Axioms [1] which is considered the former complete functional dependencies inference system.

One of the requirements of the Wastl's method is that the input must be a set of functional dependencies $X \to a$ where X is a set of attributes and a is an attribute. This kind of formulas are named *atomic functional dependencies*. The *fragmentation rule* allows transform any set of FDs in an equivalent atomic formula set

$$\{X \to Y\} \equiv \{X \to a \mid a \in Y\}$$

Although the above equivalency allows the Wastl's method to be applied to any set of FDs, it produces an important growth in the size of the input set. One of the advantages of our method is that it does not require the FDs to be atomic, providing a reduction in the tableaux.

Besides that, the $\mathbf{SL}_{FD}$ inference rules reduce the set of FDs in each branching step and so the tableaux size is decreased, pruning the search space.

Moreover, we improve our method by applying a set of algebraic results regarding keys which reduce the size of the original problem to be treated.

The paper is organized as follows: Wastl's method is presented and its execution is illustrated with an example where the tableaux is depicted level by level (Section 2). Section 3 presents the $\mathbf{SL}_{FD}$ inference system and an algebraic study used to get a more simple problem equivalent to the original one. We present in Section 4 the algorithm to find all minimal keys based on $\mathbf{SL}_{FD}$. We also provide an example to illustrate its execution and compare it with the execution of Wastl's method. We finish the paper with some conclusions and future work.

2 A tableaux approach to find keys

Due to space limitation, we refer those readers non familiar with the basic notions of functional dependency and relational databases to [3]. Let $\mathcal{A}$ be a finite set of atoms (attributes) and $\to$ be a binary connective representing a functional dependency. The language for functional dependencies is $\mathcal{L}_{FD} = \{X \to Y \mid X, Y \in 2^{\mathcal{A}}\}$. We denote the attributes with lowercase characters a, b, c, d, etc. and a general set of attributes with uppercase symbols X, Y, Z, etc. As usual, XY is used to denote the union of two sets X, Y; $X \subseteq Y$ denoted the set inclusion and $Y - X$ is used as the set difference.

The former inference system to give a notion of syntactic inference for functional dependencies is the Armstrong's Axioms (which is sound and complete). Based on it, we may also introduce the notion of key from the closure operator. Let X be a set of atoms and Γ a set of FDs, the set X^+, named the closure of X, is the maximum subset $X^+ \subseteq \mathcal{A}$ such that $X \to X^+$ can be inferred from Γ, $\Gamma \vdash X \to X^+$. A direct consequence of this definition is that $\mathcal{K} \subseteq \mathcal{A}$ is a key iff $\mathcal{K}^+ = \mathcal{A}$. Obviously, the set of all attributes in a relation constitutes a key, since $\mathcal{A}^+ = \mathcal{A}$.

The notion of minimal key is introduced to avoid the use of keys with superfluous attributes. A set of attributes $\mathcal{K} \subseteq \mathcal{A}$ is a minimal key if it is a key and there does not exists another key $\mathcal{K}' \subset \mathcal{K}$.

In this work we deal with relation schemes, i.e. pairs $< \mathcal{A}; \Gamma >$ where $\mathcal{A}$ is the set of attributes and Γ a set of FDs. In the rest of the paper we will use the term relation to refer to a relation schema when no confusion may be induced.

For simplicity without loss of generality, we consider that $\mathcal{A}$ is the set of attributes that is present in all the dependencies in Γ. Simply, if the initial set of attributes $\mathcal{A}_0$ has attributes K_0 that are not in any dependency, these attributes will be in all the minimal keys and we built $\mathcal{A} = \mathcal{A}_0 \setminus K_0$.

R. Wastl proposes a tableaux-method for deriving all the minimal keys of a relation schema using an inference system, named $\mathbb{K}$, which deals with relational schemas in the form $R =< \mathcal{A}; \{X_1 \to a_1, \ldots, X_n \to a_n\} >$ where the FDs are non-trivial (i.e. $a_i \notin X_i$) and $\mathcal{A} = \bigcup_{i=1}^{n} (X_i \cup \{a_i\})$.

The rules of the $\mathbb{K}$ inference system are the following:

$$\mathbb{K}_1 : \frac{X \to a \qquad Ya \to b}{XY \to b} \qquad \mathbb{K}_2 : \frac{X \to a \qquad Y \to b}{XY \to b}$$

This axiomatic system is not complete and it is only designed to build a tableaux as a tool to infer all minimal keys.

The tableaux T for the relation schema R is built as follows: its root is a functional dependency $X_1 \ldots X_n \to a_n$ derived from $\Gamma = \{X_1 \to a_1, X_2 \to a_2, \ldots X_n \to a_n\}$ by applying exhaustively $n - 1$ times the $\mathbb{K}_2$ rule [1]. Wastl's algorithm relies on the fact that $(X_1 \ldots X_n)^+ = \mathcal{A}$, i.e. $X_1 \ldots X_n$ is a key, and additionally, for all $\mathcal{K}$ minimal key of R we have that $\mathcal{K} \subseteq X_1 \ldots X_n$ [10].

The branches of the tableaux are built using the inference rule $\mathbb{K}_1$ as follows: let $Z \to b$ be any node in T, for each $X_i \to a_i \in \Gamma$ such that $a_i \in Z$, $(Z - a_i)X_i \to b$ is generated as a successor node and the edge between $Z \to b$ and this new child will be labeled with a_i. To avoid superfluous branches which determine a cycle, Wastl only considers in the edges those functional dependencies $X_i \to a_i$ which satisfy

$$X_i \cap L = \varnothing \tag{1}$$

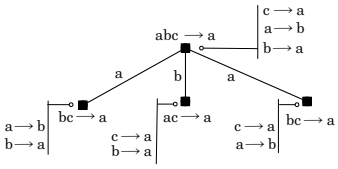
where L is the union of the edge labels on the (unique) path from the root to the node $Z \to b$. Once the tableaux has been built, in the leaves we find the keys of R, i.e. if $Y \to c$ is a leaf, then Y is a key. More specifically, the set of all the keys obtained in the leaves contains all the minimal keys [10], although it may also contain some superkeys.

Thus, the set of all minimal keys is $\uplus\{K \mid K \to a$ is a leaf of the tableaux$\}$ where the $\uplus$ operator renders the minimal elements with regard to $(2^{\mathcal{A}}, \subseteq)$ of the union.

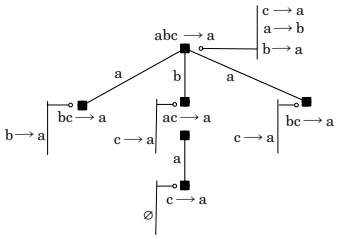
[1] In Wastl's method, the right hand side of the root is not significant and he also provides an alternative notation of the method where this singleton attribute is removed. We use the former notation because it remains faithful to the $\mathbb{K}$ system.

We show the execution of Wastl's algorithm in this illustrative example which appears in [10].

Example 2.1 *Let $\mathcal{A} = \{a, b, c\}$ and $\Gamma = \{c{\rightarrow}a, a{\rightarrow}b, b{\rightarrow}a\}$ the set of attributes and FDs respectively. We build the root of the Wastl tree ($abc{\rightarrow}a$) by applying the $\mathbb{K}_2$ rule. In the first level, by applying $\mathbb{K}_1$ we generate three branches which correspond to the three FDs in Γ* [2].



Now, we prune the Γ set in each branch according to equation (1) to avoid cycles and we apply the $\mathbb{K}_1$ rule when it holds to obtain:



As we have mentioned, every key appears in some leaf of the tableaux [10]. Here, the leaves are bc and c. We apply the $\biguplus$ union to obtain $\{c\}$ as the set of all minimal keys in $< \mathcal{A}, \Gamma >$.

We would like to remark that all the minimal keys algorithms introduced in the literature consider this operation as its last step.

3 Previous results

3.1 The Simplification Logic

The well known Armstrong's axiomatic system [1] is the former system introduced to manage FDs in a logic style. This system has influenced the design of several logics for FDs, all of them built around the transitivity paradigm. In this section, we summarize the axiomatic system of Simplification Logic for Functional Dependencies, $\mathbf{SL}_{FD}$ [6]. It avoids the use of transitivity and is guided by the idea of simplifying the set of FDs by removing redundant attributes efficiently.

$\mathbf{SL}_{FD}$ is defined as the pair $(\mathcal{L}_{FD}, \mathcal{S}_{FD})$ where $\mathcal{S}_{FD}$ has the following axiom scheme and inference rules. The third rule is named *Simplification* rule and it is the core of $\mathbf{SL}_{FD}$:

Axiom:

$$\lfloor Ax \rfloor \quad \frac{Y \subseteq X}{X{\rightarrow}Y}$$

Rules of inference:

$$\lfloor Frag \rfloor \quad \frac{X{\rightarrow}Y}{X{\rightarrow}Y'}, \quad Y' \subseteq Y \quad \lfloor Comp \rfloor \quad \frac{X{\rightarrow}Y,\ U{\rightarrow}V}{XU{\rightarrow}YV}$$

$$\lfloor Simp \rfloor \quad \frac{X{\rightarrow}Y,\ U{\rightarrow}V}{(U-Y){\rightarrow}(V-Y)}, \quad X \subseteq U, X \cap Y \neq \varnothing$$

We would like to remark that we consider general non-trivial FDs and not only non-trivial *unit* FDs as Wastl does. This is one of the strong points of our approach, because it avoids the exhaustive application of the fragmentation rule to the original FDs set, which would produce a significant growth in the size of this set.

The equivalence between $\mathbf{SL}_{FD}$ and Armstrong's Axioms and an efficient algorithm to compute the closure of a set of attributes may be seen in [6]. The authors also designed and developed a comparison with other closure algorithms in the literature to prove the practical orientation of our logic system.

One of the main advantages of $\mathbf{SL}_{FD}$is that inferences rules may be considered equivalence rules and they are enough to compute all the derivations (see [6] for further details and proofs).

The *deduction* ($\vdash$) and *equivalence* ($\equiv$) are defined as usual: We say that a FD φ is deduced from a set of FDs Γ, denoted $\Gamma \vdash \varphi$, if there exists a chain of FDs $\varphi_1 \ldots \varphi_n$ such that $\varphi_n = \varphi$ and, for all $1 \leq i \leq n$, we have that $\varphi_i \in \Gamma$, φ_i is an axiom or is obtained by applying an inference rule to the formulas in $\{\varphi_j \mid 1 \leq j < i\}$.

We say that the sets Γ and Γ' are *equivalent*, denoted $\Gamma \equiv \Gamma'$, if for all FD φ, we have that $\Gamma \vdash \varphi$ if and only if $\Gamma' \vdash \varphi$.

We introduce here the equivalence of the simplification rule, which will be used in this paper.

Proposition 3.1 *Let $R =< \mathcal{A}, \Gamma >$ be a relational schema and $X, Y, U, V \in 2^{\mathcal{A}}$ whith $X \subseteq U$. We have that:*

$$\{X{\rightarrow}Y,\ U{\rightarrow}V\} \equiv \{X{\rightarrow}Y, (U-Y){\rightarrow}(V-Y)\}$$

The language of $\mathbf{SL}_{FD}$ allows the empty set $\varnothing$ to be included in both sides of the FDs. Regarding the semantics, the formula $X \rightarrow \varnothing$ represents an axiom and the interpretation of the formula $\varnothing{\rightarrow}X$ could allows us to ensure that values of attributes X are singletons, that is, the domain for each attribute in X has only a value. Nevertheless this formula schema is introduced in $\mathbf{SL}_{FD}$ because of its powerful syntactic dimension.

In [6] an algorithm to compute closures using $\mathbf{SL}_{FD}$ is presented. In that paper, the formula $\varnothing{\rightarrow}X$ is used as a seed by the reasoning method to render the closure X^+ just by applying some operators based on the $\lfloor Simp \rfloor$ inference rule. The results presented in [6] allow us to introduce a dual expression of the FDs:

Theorem 3.2 *Let $R =< \mathcal{A}, \Gamma >$ be a relational schema and $X{\rightarrow}Y$ a functional dependency, then*

$$\Gamma \vdash X{\rightarrow}Y \quad \textit{iff} \quad \{\varnothing{\rightarrow}X\} \cup \Gamma \vdash \{\varnothing{\rightarrow}Y\}$$

3.2 Some results about keys to narrow the key finding problem

As we mention in Section 2, Wastl considers that each attribute in $\mathcal{A}$ appears at least once in a functional dependency.

[2] Observe that the first and third branches are labeled with the same attribute but they correspond to different FDs in Γ. We have preserved the original labels of R. Wastl in the edges. In our method, we will clarify the notation for a better understanding of the tableaux construction.

He ensures that those elements in $\mathcal{A}$ which does not appears in any functional dependency must be in every key. Such results are very useful to reduce the key finding problem before the method was applied. In [7] a deep study around the key notion is developed. More concretely, the authors introduce a set of theoretical results to characterize the subsets of attributes of $2^{\mathcal{A}}$ that may be minimal keys of the relation. They also designed an efficient method to reduce the key finding problem narrowing the $\mathcal{A}$ and Γ sets.

Finally, a procedure to build the minimal keys of R from the minimal keys of the reduced schema is provided. We sumarize in this section the most outstanding results introduced in [7].

Definition 3.3 *Let $R =< \mathcal{A}, \Gamma >$ be a relational schema. We define the determinant as Dnt$(\Gamma) = \bigcup_{X \to Y \in \Gamma} X$ and the determinate as Dte$(\Gamma) = \bigcup_{X \to Y \in \Gamma} Y$*

Then we define the core and the body of R as follows:

$\texttt{core} = \mathcal{A} - Dte(\Gamma)$

$\texttt{body} = (Dnt(\Gamma) \cap (\mathcal{A} - \texttt{core}^+))$

Example 3.1 *Let $R =< \mathcal{A}, \Gamma >$ where $\mathcal{A}$ is the attribute set $\{a, b, c, d, e, f\}$ and Γ is the FDs set $\{ab \to c, a \to g, g \to c, b \to h, bh \to d, c \to d, e \to f, f \to e\}$.*

$$\texttt{core} = ab \quad \textit{and} \quad \texttt{body} = ef$$

Theorem 3.4 *Let $R =< \mathcal{A}, \Gamma >$ be a scheme. Let $\mathcal{K}$ be a minimal key of R, then we have that* $\texttt{core} \subseteq \mathcal{K} \subseteq (\texttt{core} \cup \texttt{body})$.

Example 3.2 *The minimal keys of the relation R in example 3.1 are $\{abe, abf\}$. It fulfills with Theorem 3.4 which ensures that every minimal key $\mathcal{K}$ of R must satisfy $ab \subseteq \mathcal{K} \subseteq abef$.*

Example 3.3 *In example 2.1 extracted from [10] we have that* $\texttt{core} = \{c\}$ *and* $\texttt{body} = \varnothing$*. So the unique minimal key is c which arises directly with no execution of a key finding method.*

From the above results, it is possible to reduce the key finding problem for $< \mathcal{A}, \Gamma >$ to the problem of finding all minimal keys for $< \mathcal{A}', \Gamma' >$ where $\mathcal{A}' = \texttt{body}$ and $\Gamma' = \{X \cap \texttt{body} \to Y \cap \texttt{body} \mid X \to Y \in \Gamma\}$. Let $\mathcal{K}$ be a minimal key for the problem $< \mathcal{A}', \Gamma' >$, then the set $\mathcal{K} \cup \texttt{core}$ is a key for the original problem $< \mathcal{A}, \Gamma >$.

See [7] for a more complete and detailed presentation of these theoretical results and some illustrative examples.

4 A new key algorithm based on $\mathbf{SL}_{FD}$

Wastl's algorithm may be considered the first algorithm to find all minimal keys using a tableaux. Nevertheless, it is far to be considered an alternative to previous minimal key methods because it does not have a good performance. Wastl's method is suitable to find minimal keys in small examples but it cannot be executed without exceeding the machine limitations for small examples.

Our algorithm follows the Hilbert style of Wastl's algorithm but an important improvement has been achieved. Our benefits are due to three issues:

- We work with general non-trivial functional dependency, which avoid the growth in the cardinality of Γ.
- The results from Section 3.2 allow us to narrow the input and deal with a reduced version of the schema.
- We use a powerful operator derived from our simplification $\mathbf{SL}_{FD}$ rules, which reduces the search space.

The last item cited above requires the introduction of a new operator directly based on $\mathbf{SL}_{FD}$ rules.

Definition 4.1 *Let $X \to Y, U \to V$ be functional dependencies. We define the following operator:*

$$\Psi_{X \to Y}(U \to V) = \begin{cases} U \to V\text{-}Y, \textit{ if } U \cap Y = \varnothing \\ (UX)\text{-}Y \to V\text{-}(XY) \textit{ otherwise} \end{cases}$$

Let Γ be a set of functional dependencies. We generalize the Ψ operator as follows:

$$\Psi_{X \to Y}(\Gamma) = \{\Psi_{X \to Y}(U \to V) \mid U \to V \in \Gamma\}$$

We would like to remark that after the application of the Ψ operator, the right side of a functional dependency may be the empty set. In this case, it can be removed from the set because it stands for an axiom.

Example 4.1 *Let $\Gamma = \{ab \to c, cd \to a, e \to a\}$ be a set of FDs. We illustrate the application of the Ψ operator as follows:*

$$\Psi_{e \to a}(\{ab \to c, cd \to a\}) = \{be \to c, cd \to \varnothing\} = \{be \to c\}$$

In the following subsection we introduce the $\mathbf{SL}_{FD}$ algorithm to find all minimal keys

4.1 $\mathbf{SL}_{FD}$ Key Algorithm

Our algorithm is directly based on the $\mathbf{SL}_{FD}$ and more specifically it is strongly guided by the execution of the $\Psi_{X \to Y}$ operator without any heuristic tricks. It also uses the algebraic results presented in Section 3.2 to reduce the search space, providing a method to build the minimal keys for the original schema from the minimal keys obtained for the reduced schema. It is described in Algorithm 1.

Algorithm 1: $\mathbf{SL}_{FD}$ Key Algorithm

Data: $< \mathcal{A}, \Gamma >$, a relational schema.
Result: $\mathcal{L}$ the set of all minimal keys for $< \mathcal{A}, \Gamma >$.
begin

```
--Apply Section 3.2 to reduce the problem
    A' := body(A, Γ)
    Γ' := {X ∩ A' → Y ∩ A' | X → Y ∈ Γ}
--Built the tableaux and return the set of keys L'.
See algorithm 2 (Tableaux).
    L' := Tableaux(A', Γ')
--Built the set of all minimal keys L.
    L := {K ∪ core(A, Γ) | K ∈ L'}
```

As we mention in example 3.2 we cannot use the same example presented in [10]. While Waslt's method has to apply several times the $\mathbb{K}$ axiomatic system to find all minimal keys, our algorithm reduces the problem and the tableaux does not need to be built. We introduce here a more complex example to illustrate how the $\mathbf{SL}_{FD}$ algorithm works.

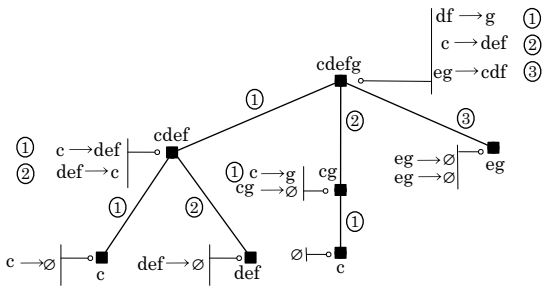


Figure 1: $\mathbf{SL}_{FD}$ Key Tableaux

Example 4.2 *Let* $\mathcal{A} = \{a, b, c, d, e, f, g\}$ *and* $\Gamma = \{adf \rightarrow g, c \rightarrow def, eg \rightarrow bcdf\}$ *a set of attributes and functional dependencies of a relational scheme R.*

We have that `core` $= \{a\}$ *and* `body` $= \{c, d, e, f, g\}$. *So, we reduce the problem having* $\mathcal{A}' = \{c, d, e, f, g\}$ *and* $\Gamma' = \{df \rightarrow g, c \rightarrow def, eg \rightarrow cdf\}$. *The* $\mathbf{SL}_{FD}$ *key tableux for this reduced problem is depicted in figure 1. We have labelled the functional dependencies in each node to be referenced in the edges. This small change simplifies the reading of the tableaux.*

As in Wastl's method, this procedure allows us to built a tableaux where the keys appears in the leaves. Notice that in each node the set of functional dependencies are treated using the Ψ *operator, which produces a great benefit in the efficiency.*

Finally we built the set of all the minimal keys for R *by adding the core (in this example no superkey has to be removed). Thus the set of all the minimal keys is* $\{ac, adef, aeg\}$.

Algorithm 2: Tableaux

Data: $< \mathcal{A}, \Gamma >$, a relational schema.
Result: $\mathcal{L}$ the set of all minimal keys for $< \mathcal{A}, \Gamma >$.
begin
  if $\Gamma = \varnothing$ **then**
    return $\mathcal{A}$
  else
    return $\biguplus_{X \rightarrow Y \in \Gamma}$ $\mathbf{Tableaux}(\mathcal{A} - Y, \Psi_{X \rightarrow Y}(\Gamma))$

As a conclusion of this illustrative example, we would like to remark that our tableaux has 7 nodes and 3 levels depth, while the same example in Wastl's method produces a tableaux of 56 nodes and 5 levels depth. Despite the benefits in the use of the Ψ *operator, the axiomatic system allows us to work with generalized formulas. Thus, we work with a* Γ *set which has 3 functional dependencies.* $\mathbb{K}$ *axiomatic system forced the formulas to be unit functional dependencies. Consequently fragmentation rule has to be applied before Wastl's method was executed, rendering a* Γ *with 8 functional dependencies.*

Besides that, notice that some minimal keys are computed several times. With our method the ac *minimal key is computed twice. In the Wastl's method, the* ac *minimal key is computed 15 times (10 times embedded in a superkey) and the* aeg *minimal key is computed 9 times. The* $adef$ *minimal key is computed only once.*

4.2 Soundness and Completeness results

In this section we introduce the theoretical foundation which ensures the correct application of our algorithm.

Proposition 4.2 *Let* Γ *be a set of functional dependencies and* $X \rightarrow Y \in \Gamma$. *We have that* $\Gamma \vdash \Psi_{X \rightarrow Y}(\Gamma)$.

Proof:
We use induction to reduce the proof to the following: Let $X \rightarrow Y, U \rightarrow V$ be functional dependencies, we have to prove that:

$$\{X \rightarrow Y, U \rightarrow V\} \vdash \Psi_{X \rightarrow Y}(U \rightarrow V)$$

If $U \cap Y = \varnothing$ we have that $U \rightarrow V \vdash U \rightarrow V - Y$ using fragmentation rule.

To prove the second case of the Ψ operator, we built the following derivation chain:

1. $X \rightarrow Y$ ..... hypothesis
2. $U \rightarrow V$ ..... hypothesis
3. $UX \rightarrow VY$ ..... 1., 2. and $\lfloor Comp \rfloor$
4. $UX - Y \rightarrow V - Y$ ..... 1., 3. and $\lfloor Simp \rfloor$
5. $UX - Y \rightarrow V - XY$ ..... $\lfloor Frag \rfloor$

Definition 4.3 *Let* Γ *be a set of functional dependencies we define its closure as:* $\Gamma^+ = \{X \rightarrow Y \mid \Gamma \vdash X \rightarrow Y\}$

The following corollary is a direct consequence of Proposition 4.2.

Corollary 4.4 *Let* Γ *be a set of functional dependencies and* $X \rightarrow Y \in \Gamma$. *We have that:* $\Psi_{X \rightarrow Y}(\Gamma)^+ \subseteq \Gamma^+$

We now present the soundness theorem of the Ψ operator regarding the finding of keys. First, we extend the Ψ operator to be applied to a relational schema and then the soundness theorem is introduced.

Definition 4.5 *Let* $R =< \mathcal{A}; \Gamma >$ *be a relation schema and* $X \rightarrow Y \in \Gamma$ *we define:* $\Psi_{X \rightarrow Y}(R) =< \mathcal{A} - Y; \Psi_{X \rightarrow Y}(\Gamma) >$

Theorem 4.6 (Soundness) *Let* $R =< \mathcal{A}; \Gamma >$ *be a relation schema and* $X \rightarrow Y \in \Gamma$ *a functional dependency, then we have that every key of* $\Psi_{X \rightarrow Y}(R)$ *is a key of R.*

Proof:
Let $R =< \mathcal{A}, \Gamma >$ be a relation schema and $X \rightarrow Y \in \Gamma$ a functional dependency and $\mathcal{K}$ a key of $\Psi_{X \rightarrow Y}(R)$.

We prove the theorem by demonstrating that $\Psi_{X \rightarrow Y}(\Gamma) \vdash \mathcal{K} \rightarrow \mathcal{A} - Y$ implies $\Gamma \vdash \mathcal{K} \rightarrow \mathcal{A}$.

From Corollary 4.4, $\Gamma \vdash \mathcal{K} \rightarrow \mathcal{A} - Y$. The following chain of derivation allows us to deduce $\Gamma \vdash \mathcal{K} \rightarrow \mathcal{A}$

1. $\mathcal{K} \rightarrow \mathcal{A} - Y$
2. $\mathcal{K} \rightarrow X$ ..... by 1 and $\lfloor Frag \rfloor$
3. $X \rightarrow Y$ ..... hypothesis
4. $\mathcal{K}X \rightarrow \mathcal{A}$ ..... by 1, 3 and $\lfloor Comp \rfloor$
5. $\mathcal{K} \rightarrow \mathcal{A} - X$ ..... by 2, 4 and $\lfloor Simp \rfloor$
6. $\mathcal{K} \rightarrow \mathcal{A}$ ..... by 2, 5 and $\lfloor Comp \rfloor$

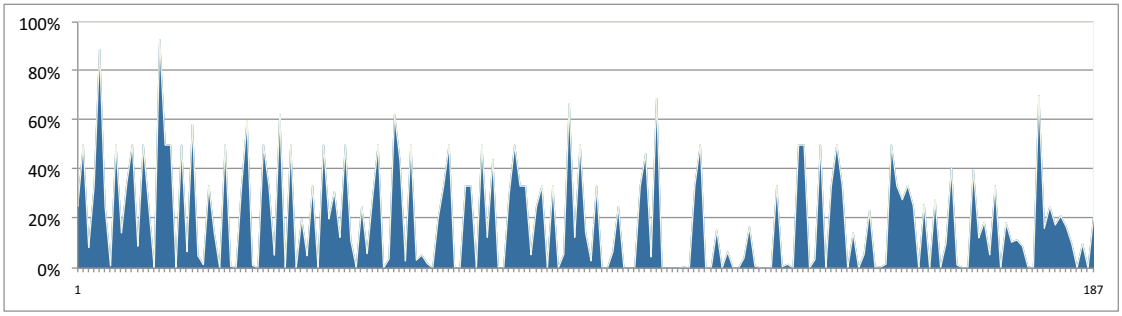


Figure 2: Percentage of $\mathbf{SL}_{FD}$ w.r.t Wastl method.

The proof of the completeness theorem makes use of the point of view of FDs induced by the $\mathbf{SL}_{FD}$ presented in Theorem 3.2. First, we enunciate the following lemma (the proof is straightforward from Theorem 3.2):

Lemma 4.7 *Let $R =< \mathcal{A}; \Gamma >$ be a relation schema. $\mathcal{K}$ is key in R iff $\{\varnothing \rightarrow \mathcal{K}\} \cup \Gamma \equiv \{\varnothing \rightarrow \mathcal{A}\}$.*

Proposition 4.8 *Let $R =< \mathcal{A}, \Gamma >$ be a relation schema and $X \rightarrow Y \in \Gamma$ a functional dependency. If $\mathcal{K}$ is a key of R and $X \subseteq \mathcal{K}$ then $\mathcal{K}$ is a key of $\Psi_{X \rightarrow Y}(R)$.*

Proof:
Like in proof of Theorem 4.2 we may reduce this proof to the following: let $\mathcal{K}, X, Y, U, V \in 2^{\mathcal{A}}$ we have to prove that $\{\varnothing \rightarrow \mathcal{K}, X \rightarrow Y, U \rightarrow V\} \equiv \{\varnothing \rightarrow \mathcal{K}, X \rightarrow Y, \Psi_{X \rightarrow Y}(U \rightarrow V)\}$.

From $\lfloor Simp \rfloor$ equivalence (Proposition 3.1) we have that $\{\varnothing \rightarrow \mathcal{K}, X \rightarrow Y, U \rightarrow V\} \equiv \{\varnothing \rightarrow \mathcal{K}Y, U - \mathcal{K}Y \rightarrow V - \mathcal{K}Y\}$. We tackle the problem separately according to the result of the intersection $Y \cap U$ which determines the two cases in the detifition of the Ψ operator.

If $Y \cap U = \varnothing$ then

$$\begin{array}{ll} \{\varnothing \rightarrow \mathcal{K}, X \rightarrow Y, \Psi_{X \rightarrow Y}(U \rightarrow V)\} & \overset{\Psi def.}{=} \\ \{\varnothing \rightarrow \mathcal{K}Y, U \rightarrow V - Y\} & \overset{\lfloor Simp \rfloor}{=} \\ \{\varnothing \rightarrow \mathcal{K}Y, U - \mathcal{K}Y \rightarrow (V - Y) - \mathcal{K}Y\} & = \\ \{\varnothing \rightarrow \mathcal{K}Y, U - \mathcal{K}Y \rightarrow V - \mathcal{K}Y\}. & \end{array}$$

If $Y \cap U \neq \varnothing$ then

$$\begin{array}{ll} \{\varnothing \rightarrow \mathcal{K}, X \rightarrow Y, \Psi_{X \rightarrow Y}(U \rightarrow V)\} & \overset{\Psi def.}{=} \\ \{\varnothing \rightarrow \mathcal{K}Y, UX - Y \rightarrow V - XY\} & \overset{\lfloor Simp \rfloor}{=} \\ \{\varnothing \rightarrow \mathcal{K}Y, UX - \mathcal{K}Y \rightarrow V - \mathcal{K}XY\} & \end{array}$$

We have that $UX - \mathcal{K}Y = (U \cup X) \cap \overline{\mathcal{K}} \cap \overline{Y} = (U \cap \overline{\mathcal{K}} \cap \overline{Y}) \cup (X \cap \overline{\mathcal{K}} \cap \overline{Y}) = U - \mathcal{K}Y$ and $V - \mathcal{K}XY = V \cap (\overline{\mathcal{K} \cup X \cup Y}) = V \cap (\overline{\mathcal{K} \cup Y}) = V - \mathcal{K}Y$.

We conclude this section with the Completeness Theorem:

Theorem 4.9 (Completeness) *Algorithm 1 computes all the minimal keys of an input relational schema $R =< \mathcal{A}, \Gamma >$.*

Proof:
It is a direct consequence from Proposition 4.8 and the fact that for all $\mathcal{K}$ key of R, there exists a functional dependency $X \rightarrow Y \in \Gamma$ such that $X \subseteq \mathcal{K}$. From Proposition 4.8, $\mathcal{K}$ is computed in the tableaux branch given by $\Psi_{X \rightarrow Y}$ □

We have designed an experiment to show the benefits in the use of the $\mathbf{SL}_{FD}$ tableaux. We have randomly generated 187 sets of functional dependencies. Each set has a maximum of 10 functional dependencies defined over a set of 10 attributes. These hard boundaries are due to Wastl computation behavior, which quickly becomes intractable with bigger sets of dependencies. In Figure 2 we show the percentage of the $\mathbf{SL}_{FD}$ time execution of each set of functional dependencies with respect to Wastl method. Notice that in all the cases the execution time of $\mathbf{SL}_{FD}$ method is lower than Wastl one.

For each set of dependencies, we define its size as the total number of attributes contained in the dependencies of the set. In the experiment the size ranges from 22 to 95 attributes. There does not exist a strong correlation between the size and the execution time. The average of $\mathbf{SL}_{FD}$ execution time is 601.88 ms while the average of Wastl method is 4796.01 ms.

5 Conclusions and future work

In this work we present a new algorithm to find all minimal keys in a relation scheme. We have developed an algorithm directly connected with a functional dependency logic. It follows the approach initiated by R. Wastl. As far as we know, there does not exist other tableaux based approach which follows Wastl line. We improve the Wastl's method as follows:

- Our method deals with general non-trivial dependencies which avoids an explosive growth of the input set of dependencies.
- It reduces the original problem into an equivalent and simpler one by using some theoretical result about keys.
- The use of powerful operators based on simplification rules provides a great pruning of the tableaux.

Our next step will be to make a deeper comparison of our method with Wastl method and other classical methods which appear in the literature, which empirically quantify the improvement. We have the intention to generate two versions of the algorithm: a sequential and a parallel one. We think that the tableaux paradigm will ease the design of the parallel version of the algorithm. We also intend to develop a theoretical study of the improvement of our algorithm w.r.t Wastl method.

References

[1] W. Armstrong, Dependency structures of data base relationships, in: IFIP Congress, 1974, pp. 580–583.

[2] E. F. Codd, A relational model of data for large shared data banks, Commun. ACM, 26 (1): 64–69, 1983.

[3] R. Elmasri, S. Navathe, Fundamentals of Database Systems, 6th ed. Addison-Wesley, 2011.

[4] R. Fadous, J. Forsyth, Finding candidate keys for relational data bases. SIGMOD international conference on Management of data, ACM, 1975, pp. 203–210.

[5] C. L. Lucchesi, S. L. Osborn, Candidate Keys for Relations, in: Journal of Computer and System Sciences (JCSS) 17(2):270 279, 1978.

[6] A. Mora, P. Cordero, M. Enciso, I. Fortes, G. Aguilera, Closure via functional dependence simplification, International Journal of Computer Mathematics. Volume 89, Issue 4, March 2012, pages 510-526 (2012).

[7] A. Mora, I. Pérez de Guzmán, M. Enciso, P. Cordero, Ideal non-deterministic operators as a formal framework to reduce the key finding problem, International Journal of Computer Mathematics 88 (2011) 1860–1868.

[8] H. Saiedian, T. Spencer, An efficient algorithm to compute the candidate keys of a relational database schema, The Computer Journal 39 (2) (1996) 124–132.

[9] Y. Sismanis, P. Brown, P. J. Haas, B. Reinwald, GORDIAN: efficient and scalable discovery of composite keys. Conf. on Very large databases, 2006, pp. 691–702.

[10] R. Wastl, Linear derivations for keys of a database relation schema, Journal of Universal Computer Science 4 (12) (1998) 883–897.

[11] C. T. Yu, D. T. Johnson, On the complexity of finding the set of candidate keys for a given set of functional dependencies, Inform. Process. Letters (1976) 100–101.

[12] Y. Zhang, Determining all candidate keys based on karnaugh map. IEEE International Conference on Information Management, Innovation Management and Industrial Engineering - Volume 04, 2009, pp. 226–229.