An automated reasoning method to solve the minimal key finding problem

(Submitted to Information Processing Letters)

Angel Mora Bonilla
Department of Applied Mathematics
University of Malaga, Spain

DAMOL, Palacky University Olomouc, June 2012

Table of Contents

$\mathbf{SL}_{FD}$-logic

Stages:

- Algebraic formalization of f-family and by hand for functional dependencies.
- Firstly, we proposed a new Simplification Rule adequate to remove redundancy in an automatic way.
- Simplification Rule turned the *heart* of a novel logic : $\mathbf{SL}_{FD}$ logic - Simplification logic for FDs.
- $\mathbf{SL}_{FD}$ logic turned out to be the *engine* of automated methods: redundancy removal, closure algorithm, **minimal keys**, etc.

Simplification Logic

SL$_{FD}$ *Logic: Elimination of data redundancy in knowledge representation, P. Cordero et.al.,* LNAI, *2527, pp, 141-150, 2002*

$\lfloor Axiom \rfloor$: $\vdash_{\mathcal{S}_{FD}} X \mapsto Y$, si $Y \subseteq X$

- $\lfloor Frag \rfloor$ $X \mapsto Y \vdash_{\mathcal{S}_{FD}} X \mapsto Y'$ if $Y' \subseteq Y$ .... **Fragmentation**
- $\lfloor Comp \rfloor$ $X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FD}} XU \mapsto YV$ .... **Composition**
- $\lfloor Simp \rfloor$ $X \mapsto Y,\ U \mapsto V \vdash_{\mathcal{S}_{FD}} (U\text{-}Y) \mapsto (V\text{-}Y)$ .... **Simplification**
  if $X \subseteq U$, $X \cap Y = \varnothing$

and the following derived rule:
$\lfloor rSimp \rfloor$ $X \mapsto Y, U \mapsto V \vdash_{\mathcal{S}_{FDS}} U \mapsto (V\text{-}Y)$ .... **r-Simplification**
if $X \subseteq UV, X \cap Y = \varnothing$

Table of Contents

SL_{FD} closure

Closure via functional dependence simplification, A. Mora et.al., IJCM, 89 (4), 2012

- We present an automated method directly based on Simplification Logic to calculate the closure of a set of attributes.
- Fields of application goes from theoretical areas as algebra or geometry to practical areas as Databases, Formal Concept Analysis and Artificial Intelligence: *data analysis, knowledge structures, knowledge compilation, redundant constraint elimination, query optimization, finding key problem,etc.*

$\mathbf{SL}_{\mathrm{FD}}$ closure

Theorem

- **Equivalency I**: If $U \subseteq W$ then $\{\top \mapsto W, U \mapsto V\} \equiv_{\mathcal{S}_{\mathrm{FD}}} \{\top \mapsto WV\}$
- **Equivalency II**: If $V \subseteq W$ then $\{\top \mapsto W, U \mapsto V\} \equiv_{\mathcal{S}_{\mathrm{FD}}} \{\top \mapsto W\}$
- **Equivalency III**: If $U \cap W \neq \varnothing$ or $V \cap W \neq \varnothing$ then
$$\{\top \mapsto W, U \mapsto V\} \equiv_{\mathcal{S}_{\mathrm{FD}}} \{\top \mapsto W, U - W \mapsto V - W\}$$

Automated Prover to obtain the closure

From Γ and X, calculate X^+ (the closure of X):

- Add $\top \mapsto X$
- Apply systematically the three equivalences based on $\mathbf{SL}_{\mathrm{FD}}$ logic.

Result: $\top \mapsto X^+$

Table of Contents

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Primary keys and Foreigns keys are dependencies

NUMBER	NAME	PHN	DPT
8397	Manuel Pérez	3309	133
5688	Juana Gómez	1324	133
5670	Román García	5633	38

employee

NUMBER	NAME	LOCATION
133	Sales	Central
38	Marketing	Suc-1

departament

Keys and Functional Dependencies

Normalization

To avoid inconsistencies and redundancy.

	Subject	Identity Card	Surname	Name	Closed Call
t1	Algebra	22222222A	SMITH	RALPH	4
t2	Algebra	33333333A	ROSE	PETER	1
t3	Calculus	22222222A	SMITH	RALPH	4
t4	Calculus	44444444B	BRANDON	ANNE	5
t5	Calculus	11111111C	BUGLE	LOUISE	3
t6	Numerical Methods	33333333A	ROSE	PETER	1

Keys and Functional Dependencies

Normalization

Identity Card is the key.

	Identity Card	Surname	Name	Closed Call
t1	22222222A	SMITH	RALPH	4
t2	33333333A	ROSE	PETER	1
t4	44444444B	BRANDON	ANNE	5
t5	11111111C	BUGLE	LOUISE	3

	Identity Card	Subject
t1	22222222A	Algebra
t3	22222222A	Calculus
t2	33333333A	Algebra
t6	33333333A	Numerical Methods
t4	44444444B	Calculus
t5	11111111C	Calculus

Minimal Keys: regarding to the FDs

Definition: Key

The functional dependency allows us to define the key of a relation *R* as a subset of their attributes $\mathcal{K} \subseteq \mathcal{A}$ such that the functional dependency $\mathcal{K} \mapsto \mathcal{A}$ holds.

Definition: Key

$\mathcal{K} \subseteq \mathcal{A}$ is a key iff $\mathcal{K}^+ = \mathcal{A}$.

- We may affirm that the set of all attributes in a relation constitutes a key, since $\mathcal{A}^+ = \mathcal{A}$.
- 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 $\mathcal{K}' \subset \mathcal{K}$.

Minimal Keys: regarding to the tuples

Definition: Key

Let R be a relation and $\mathcal{A}$ a set of attributes in a relational scheme. $\mathcal{K} \subseteq \mathcal{A}$ is a key if for all two tuples t_1, t_2 of R, $t_1[\mathcal{K}] \neq t_2[\mathcal{K}]$.

Really, this definition is based on the previous definition using FDs and closures. *Database books say: "no tuples must be repeated".*

Minimal Keys: regarding to the tuples

Definition: Key

Let R be a relation and $\mathcal{A}$ a set of attributes in a relational scheme. $\mathcal{K} \subseteq \mathcal{A}$ is a key if for all two tuples t_1, t_2 of R, $t_1[\mathcal{K}] \neq t_2[\mathcal{K}]$.

Really, this definition is based on the previous definition using FDs and closures. *Database books say: "no tuples must be repeated".*

Minimal Keys: two approaches

- Finding minimal keys from a set of FDs and a set of attributes (scheme of a relation): Classical finding key problem.
- Finding minimal keys from a table (a instance of a relation): Data mining.

Interest of keys

- The notion of key is one of the mainstay in the Codd's Relational Model.
- Tables need to have a primary key to fulfill Codd's integrity rules: First and Second Integrity Rules of the relational model are based on the notion of Primary Key.

Applications of keys

- Normalization (keys and 3NF).
- Data query and management.
- Data modeling.
- Query optimization.
- Indexing.
- Anomaly detection.
- Data integration

First algorithms about keys

- Delobel and Casey [Delobel, 1973] proposed the first algorithm for the finding key problem.
- Keys were studied within the framework of the implication matrix in [Fadous,1975].
- Bernstein in his Ph.D [Bernsteing, 1975] proposed probably the first usually cited algorithm to find all keys.
- Algorithm of Lucchesi and Osborn [Luchessi, Osborn, 1978] 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.
- Kundu [Kundu, 1985] proposed an algorithm for finding a single key.
- Demetrovics and Thuan [Demetrovics, 1986] describe an algorithm to find all keys which good results.
- Elmasri and Navathe [Elmasri, 1994] showed also an algorithm for finding a key.

First algorithms about keys

All the classical algorithms use the closure operator to check if a given subset of attributes is a key with regard to a set of functional dependencies.

Other paradigms:

- Saiedian and Spencer [Saiedian, 1996] propose an algorithm for computing the candidates keys using attribute graphs when it is not strongly connected.
- Wastl [Wastl, 1998] introduces a Hilbert style inference system, called $\mathbb{K}$, for deriving all keys. Wastl builds a tableaux which represents the search space to find all the keys applying the inference system $\mathbb{K}$.
- Zhang [Zhang, 2009] use Karnaugh maps to calculate all the keys.

As far as we know, Wastl Algorithm is the first approach that use inference rules to tackle the finding key problem.

First algorithms about keys

All the classical algorithms use the closure operator to check if a given subset of attributes is a key with regard to a set of functional dependencies.

Other paradigms:

- Saiedian and Spencer [Saiedian, 1996] propose an algorithm for computing the candidates keys using attribute graphs when it is not strongly connected.
- Wastl [Wastl, 1998] introduces a Hilbert style inference system, called $\mathbb{K}$, for deriving all keys. Wastl builds a tableaux which represents the search space to find all the keys applying the inference system $\mathbb{K}$.
- Zhang [Zhang, 2009] use Karnaugh maps to calculate all the keys.

As far as we know, Wastl Algorithm is the first approach that use inference rules to tackle the finding key problem.

Complexity

The problem of finding all of the keys of a relation has been shown to be NP-complete [Lucchesi, S. Osborne, 1978], [Jou and Fischer, 1982].

- Osborn shows in her Ph.D. 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'*.
- Yu and Johnson [Yu, 1976] stablished 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.
- K. Tichler establishes in [Tichler, 2004] a bound for the size of a Sperner system representing a set of minimal keys.

Usefulness of keys

- A. Sali [Sali, 2004] studies keys in higher-order datamodels and introduces an ordering between key sets, and investigates systems of minimal keys.
- Hartmann et.al. [Hartmann, 2006] present polynomial-time algorithms to determine non-redundant covers of sets of functional dependencies, and to decide whether a given set of subattributes forms a superkey.
- Hamrouni [Hamrouni, 2007] states that “minimal generators, aka minimal keys, play an important role in many theoretical and practical problem settings involving closure systems that originate in graph theory, relational database design, data mining, etc”.
- Katona et.al. [Katona, 2008] affirms “arguably, the most important database constraint is the collection of functional dependencies that instances of a relational schema satisfy, in particular the key dependencies”.

Table of Contents

Wastl Method

- R. Wastl builds a tableaux using a Hilbert style inference system, called $\mathbb{K}$.
- This axiomatic system is not complete and it is only designed to build a tableaux as a tool to infer all minimal keys.

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

Rules of inference:

$$\mathbb{K}_1 : \quad \frac{X\mapsto a \qquad Ya\mapsto b}{XY\mapsto b}$$

$$\mathbb{K}_2 : \quad \frac{X\mapsto a \qquad Y\mapsto b}{XY\mapsto b}$$

Wastl's algorithm relies on the fact that $(X_1 \dots X_n)^+ = \mathcal{A}$, i.e. $X_1 \dots X_n$ is a key, and additionally, for all $\mathcal{K}$ minimal key of R we have that $\mathcal{K} \subseteq X_1 \dots X_n$

Wastl Method

- The tableaux represents the search space to find all the keys.
- Each step in the tableaux construction is guided by the application of the inference system $\mathbb{K}$.

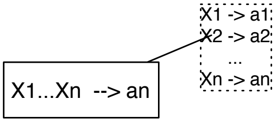
Tableaux

- Root is a functional dependency $X_1 \ldots X_n \mapsto a_n$ derived from $\Gamma = \{X_1 \mapsto a_1, X_2 \mapsto a_2, \ldots X_n \mapsto a_n\}$ **($\mathbb{K}_2$ rule).**
- Each step in the tableaux construction is guided by the application of **($\mathbb{K}_1$ rule).**

Wastl Method

Tableaux

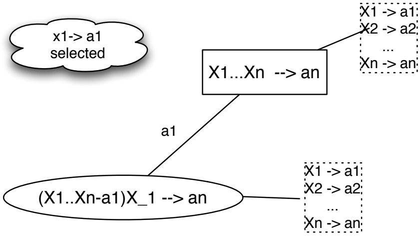
- **[Step1]** Root is a functional dependency $X_1 \ldots X_n \mapsto a_n$ derived from $\Gamma = \{X_1 \mapsto a_1, X_2 \mapsto a_2, \ldots X_n \mapsto a_n\}$ **($\mathbb{K}_2$ rule).**



Wastl Method

Tableaux: each step applying of ($\mathbb{K}_1$ rule)

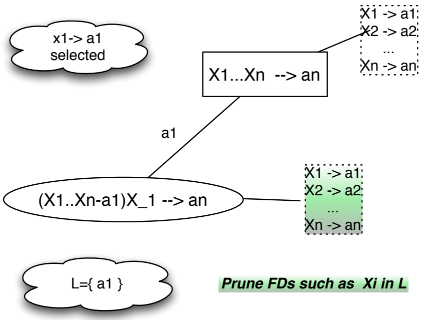
- **[Step2]** Let $X_1 \dots X_n \mapsto b$ be any node in T, for each $X_i \mapsto a_i \in \Gamma$ such that $a_i \in X_1 \dots X_n$, $(X_1 \dots X_n - a_i)X_i \mapsto b$ is generated as a successor node and the edge between $X_1 \dots X_n \mapsto b$ and this new child will be labeled with a_i.



Wastl Method

Tableaux: Applying $\mathbb{K}_1$ rule

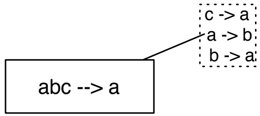
- **[Step2]** To avoid superfluous branches which determine a cycle, Wastl only considers in the edges those FDs $X_i \mapsto a_i$ which satisfy $Xi \cap L = \varnothing$ (where L is the union of the edge labels on the (unique) path from the root to the node $Z \mapsto b$).



Wastl Method

Example: Step 1

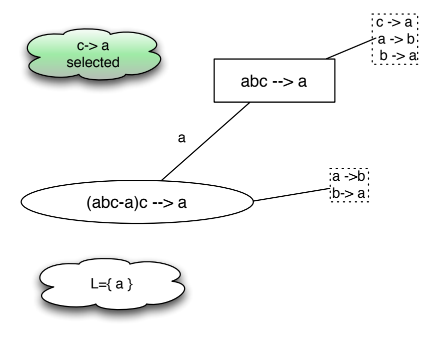
Let $\mathcal{A} = \{a, b, c\}$ and $\Gamma = \{c{\mapsto}a, a{\mapsto}b, b{\mapsto}a\}$. We build the root of the Wastl tree ($abc{\mapsto}a$) by applying the $\mathbb{K}_2$ rule.



Wastl Method

Example: Step 2

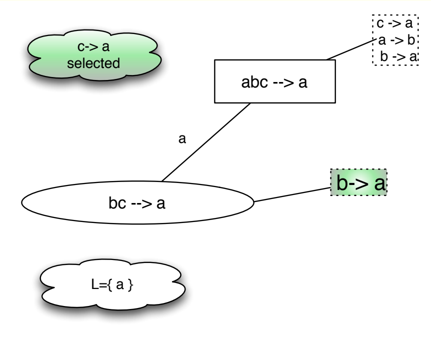
And applying $\mathbb{K}_1$ we build the tableaux.



Wastl Method

Example: Step 3

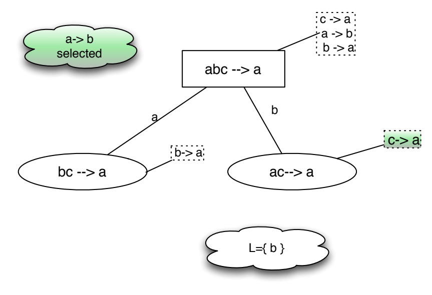
Pruning the dependencies.



Wastl Method

Example: Step 2, Step 3

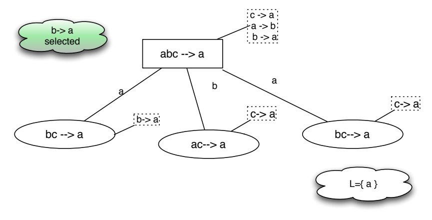
Applying $\mathbb{K}_1$ for other FD of the root, etc.



Wastl Method

Example: Step 2, Step 3

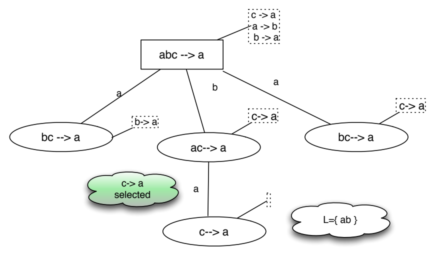
Applying $\mathbb{K}_1$ for other FD of the root, etc.



Wastl Method

Example: Step 2, Step 3

Finally, the tableaux is:



Wastl Method

Example: Step 2, Step 3

All the keys appears at least once in one tableaux leaf. Here, the leaves are *bc* and *c*. We apply the $\uplus$ union to obtain $\{c\}$ as the set of all minimal keys in $< \mathcal{A}, \Gamma >$.

All the minimal keys algorithms introduced in the literature consider this operation as its last step.

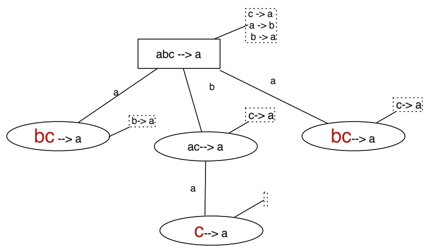


Table of Contents

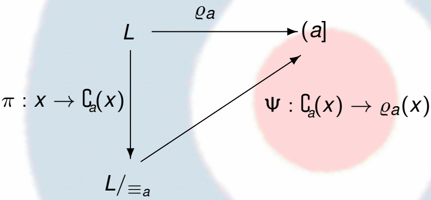
Pruning the search space for keys

***Ideal non-deterministic operators as a formal framework to reduce the key finding problem, A. Mora et. al.*, IJCM, 88 (9), 2011**

- We have presented a formal method in the framework of the lattice theory to prune the problem of finding all the minimal keys.
- With lineal cost, this prune method provides a longer reduction than the rest of techniques (The %-reduction in an experiment was the 70,52 %).

We define $\varrho_a : A \to (a]$ with $\varrho_a(x) = x \wedge a$

- $((a], \leq)$ defines a Boole Algebra
- $\pi : L \to L/_{\equiv_a}$ is the homomorphism that assigns to x its equivalence class $\mathbb{C}_a(x)$
- $\Psi : L/_{\equiv_a} \to (a]$ is the isomorphism defined as $\Psi(\mathbb{C}_a(x)) = \varrho_a(x)$



Prunning the scheme

Algorithm: *core* and the *body* of R

Let $R =< \mathcal{A}, \Gamma >$ be a relational schema.

1. $Dnt(\Gamma) = \bigcup_{X \mapsto Y \in \Gamma} X$
2. $Dte(\Gamma) = \bigcup_{X \mapsto Y \in \Gamma} Y$
3. core $= \mathcal{A} - Dte(\Gamma)$
4. body $= (Dnt(\Gamma) \cap (\mathcal{A} - core^+))$

Theorem

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

Prunning the scheme

Example

Let $\mathcal{A} = \{a, b, c, d, e, f, g\}$ and $\Gamma = \{adf\mapsto g, c\mapsto def, eg\mapsto bcdf\}$.

1. $Dnt(\Gamma) = \bigcup_{X\mapsto Y\in\Gamma} X = \{a, c, d, e, f, g\}$
2. $Dte(\Gamma) = \bigcup_{X\mapsto Y\in\Gamma} Y = \{b, c, d, e, f, g\}$
3. core $= \mathcal{A} - Dte(\Gamma) = \{a\}$
4. body $= (Dnt(\Gamma) \cap (\mathcal{A} - core^{+})) = \{c, d, e, f, g\}$

So, we reduce the problem considering

$\mathcal{A}' = \{c, d, e, f, g\}$ and $\Gamma' = \{df\mapsto g, c\mapsto def, eg\mapsto cdf\}$.

Table of Contents

SL_{FD}-Key Algorithm

Automated reasoning to infer all minimal keys, P. Cordero et.al., Submitted.

We define Ψ operator directly based on $\mathbf{SL}_{FD}$-logic. We construct the tableaux in a similar way.

Definition: Ψ-Operator

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

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

SL_{FD}-Key Algorithm

SL_{FD}-Key Algorithm follows the Hilbert style of Wastl's Algorithm but an important improvement has been achieved.

Improvements with respect to Wastl's Algorithm

- We work with general non-trivial functional dependency, which avoids the growth in the cardinal of Γ.
- Prunning method of the scheme render an important reduction of the set of attributes and the set of FDs.
- The new operator Ψ derived from our simplification $\mathbf{SL}_{FD}$ rules which reduces the set of FDs in each edge and provides an improvement in the performance of the method.

SL_{FD}-Key Algorithm

Example: Pruning the scheme

Let $\mathcal{A} = \{a, b, c, d, e, f, g\}$ and $\Gamma = \{adf \mapsto g, c \mapsto def, eg \mapsto bcdf\}$.

We have that $\texttt{core}_F = \{a\}$ and $\texttt{body}_F = \{c, d, e, f, g\}$.

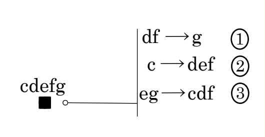
As we have shown, we reduce the problem considering

$\mathcal{A}' = \{c, d, e, f, g\}$ and $\Gamma' = \{df \mapsto g, c \mapsto def, eg \mapsto cdf\}$.

SL_{FD}-Key Algorithm

Example: Building the root

Considering $\mathcal{A}' = \{c, d, e, f, g\}$ and $\Gamma' = \{df{\mapsto}g, c{\mapsto}def, eg{\mapsto}cdf\}$.

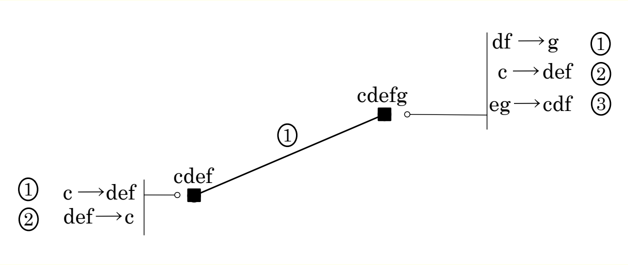


SL_{FD}-Key Algorithm

Example: Ψ – *operator*

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

- Simplifying the root *cdefg* using $df \mapsto g$.
- Simplifying the FDs: $\Psi_{df \mapsto g}(c \mapsto def) = c \mapsto def$
- Simplifying the FDs: $\Psi_{df \mapsto g}(eg \mapsto cdf) = (dfeg) - g \mapsto cdf - dfg = dfe \mapsto c$

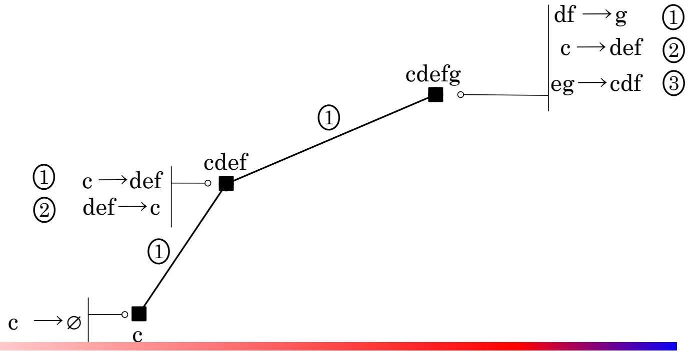


SL_{FD}-Key Algorithm

Example: Ψ – *operator*

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

- Simplifying the node *cdef* using $c\mapsto def$.
- Simplifying the FDs: $\Psi_{c\mapsto def}(def\mapsto c) = (dfec) - def\mapsto c - cdef = c\mapsto\varnothing$

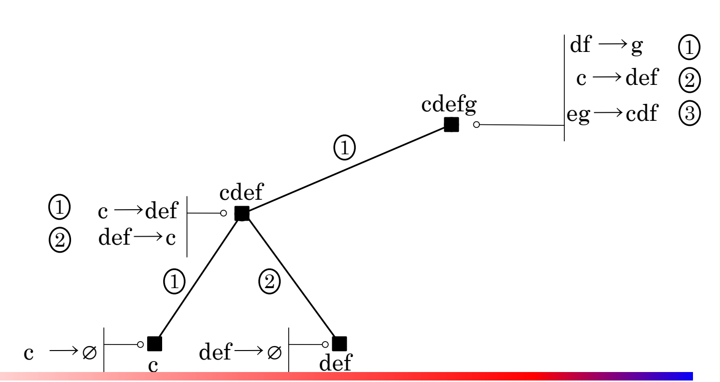


SL_{FD}-Key Algorithm

Example: Ψ – *operator*

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

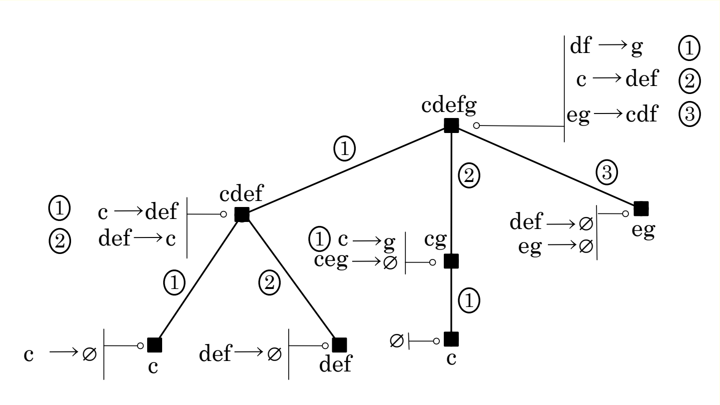
- Simplifying the node *cdef* using $def \mapsto c$.
- Simplifying the FDs: $\Psi_{def \mapsto c}(c \mapsto def) = (cdef) - c \mapsto def - defc = def \mapsto \varnothing$



SL_{FD}-Key Algorithm

Example: Ψ − *operator*

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

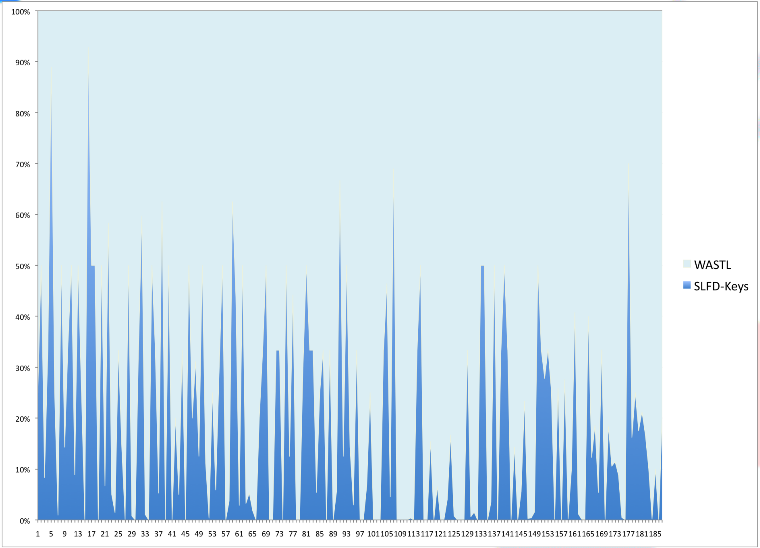


Execution

Results:

- Keys in our tableaux are $\{c, def, eg\}$
- $core = \{a\}$
- Thus the set of all the minimal keys is $\{ac, adef, aeg\}$.
- Our tableaux has 7 nodes and 3 levels of depth, while this same example in Wastl's method produces a tableaux of 56 nodes and 5 levels of depth.

Comparison: SL_{FD}-Keys versus Wastl



Comparison: SL_{FD}-Keys versus others

```
   c,d |--> b,e,i,j
   c |--> a,h,i,j
   e,j |--> a,e,h
   b,c |--> a,c,e,i
   i |--> f,g,h,i
   b |--> b,c,e,f,g,i,j
   d |--> c,h
   d |--> j
   d |--> a,b,g,h
   b |--> e
OsbornLucchesi: Comienzo de ejecución
Conjunto de claves: [d]
OsbornLucchesi: Finalización de ejecución
SLfdSimplify: Comienzo de ejecución
Proceso de eliminación de atributos en Y si: Para toda X → Y ∈ Γ | X ∩ Y != ø
Conjunto de claves: [d]
SLfdSimplify: Finalización de ejecución
Verificación "checkAlgorithms": OK
SLfdSubstitute: Comienzo de ejecución
Conjunto de claves: [d]
SLfdSubstitute: Finalización de ejecución
Verificación "checkAlgorithms": OK
SaiedianSpencer: Comienzo de ejecución
Conjunto de claves: [d]
Conjunto de claves: [d]
SaiedianSpencer: Finalización de ejecución
Verificación "checkAlgorithms": OK
Wastl: Comienzo de ejecución
Proceso de eliminación de atributos en Y si: Para toda X → Y ∈ Γ | X ∩ Y != ø
Número de DFs canónicas : 30
Conjunto de claves: [d]
Wastl: Finalización de ejecución
```

```
Estadisticas:
  Parámetros de ejecución:
  ------------------------
    Nº DFs...............: 10
    Nº Attributos........: 10
    Size.................: 47

  Resultados de ejecución
  ------------------------
            Algoritmo        T. Ejecución
    ===============   ===============
       OsbornLucchesi                 10
         SLfdSimplify                  1
        SLfdSubstitute                 1
      SaiedianSpencer                 22
                 Wastl               273
```

Comparison: SL_{FD}-Keys versus others

```
d |--> b,g,j
c,i |--> b,c,e,i,j
i |--> c,f,g
g |--> a,e,h,i,j
i |--> c,e,f,g,h
i,j |--> c,e,f,h,j
b,d |--> g
d,i |--> e,j
d,e |--> g
d |--> e,g,j
```

Algoritmo	T. Ejecución
OsbornLucchesi	8
SLfdSimplify	1
SLfdSubstitute	1
SaiedianSpencer	20
Wastl	4035

Comparison: SL_{FD}-Keys versus others

```
d |--> b,g,j
c,i |--> b,c,e,i,j
i |--> c,f,g
g |--> a,e,h,i,j
i |--> c,e,f,g,h
i,j |--> c,e,f,h,j
b,d |--> g
d,i |--> e,j
d,e |--> g
d |--> e,g,i
```

```
Resultados de ejecución
------------------------
        Algoritmo      T. Ejecución
  ===============   ===============
   OsbornLucchesi                 8
      SLfdSimplify                1
    SLfdSubstitute                1
  SaiedianSpencer                20
            Wastl              4035

  Claves mínimas......: [d]
```

Comparison: SL_{FD}-Keys versus others

```
g |--> d,f
g |--> b,e,f,j
c,g |--> e,j
g,i |--> d,e,f
f |--> a,c,e,f,g
a,h |--> j
g,i |--> b,j
b |--> c,e,f,h
c,g |--> e,f,h,j
a |--> e,f,i
```

```
Resultados de ejecución
-----------------------
        Algoritmo      T. Ejecución
  ===============   ===============
   OsbornLucchesi                21
     SLfdSimplify                22
    SLfdSubstitute                3
  SaiedianSpencer                14
            Wastl               132

  Claves mínimas......: [g,i;a,i;b,i;f,i]
```

Conclusions

Our method improves the one proposed by Wastl as follows:

- Our method deals with general non-trivial FDs.
- Our pruning method reduces the original problem into an equivalent and simpler one by using some algebraic theoretical result about keys.
- The use of powerful operator based on simplification rules provides a great pruning of the tableaux with a great reduction in the execution of the method.

Our next step will be to make a deeper comparison of our method with other classical method which appear in the literature.

Thanks


1
Birth
Form question in your mind
2
Evaluate
Is it a reasonable question?
3
Remember
Until you can ask the question
4
Courage
To ask the question out loud

An automated reasoning method to solve the minimal key finding problem

(Submitted to Information Processing Letters)

Angel Mora Bonilla
Department of Applied Mathematics
University of Malaga, Spain

DAMOL, Palacky University Olomouc, June 2012