関係データマイニングにおける頻出飽和パターン枚挙アルゴリズムについて

長野　真也　　本多　祐也　　世木　博久

名古屋工業大学　情報工学科

On Enumerating Frequent Closed Patterns in Multi-Relational Data Mining

Shinya Nagano　　Yuya Honda　　Hirohisa Seki

Department of Computer Science, Nagoya Institute of Technology

1　はじめに

関係データマイニング (multi-relational data mining : MRDM) とは、従来のデータマイニングとは異なり、述語論理式で表現されたパターン (連言) を扱い、また複数の関係表を用いたデータマイニングである．それにより、対象の持つ様々な属性や対象間の関係を表現できる高い記述能力を持っている．

MRDM でのアルゴリズムとして、Garriga らが提案した RelLCM2 [4] がある．これは，従来の DM のための効率的な頻出飽和パターン枚挙アルゴリズムである LCM [7, 8] アルゴリズムに基づき，それを論理パターンが効率的に扱えるように拡張したものである．

MRDM では通常，キー (目標概念) の概念を導入してパターンをマイニングする．例えば，WARMR [1] では、注目するアトム (キー) を指定することで，ユーザが興味のあるパターンにのみ注目することができると同時に，パターンの頻度を定義することができる．しかし，RelLCM2 ではキーの概念を導入していないため、全ての頻出飽和パターンを枚挙している．

本研究では，従来研究である RelLCM2 に基づき，それにキーの概念を導入した頻出飽和パターン枚挙アルゴリズムを提案する．キーの概念を導入することで，キーを考慮した出現集合や飽和連言の概念を定義する．それにより，飽和連言の枚挙方法に RelLCM2 の場合と比較してどのような相違があるかについて考察する．また，キーを考慮した飽和連言を効率よく枚挙するために，リテラル (アトム) 集合に順序関係を導入し，その順序を用いて探索を制御する方法を提案する．このようなキーの概念を導入した頻出飽和連言枚挙アルゴリズム *ffLCM* について説明し，その正当性について議論する．

第 2 節で，準備として関係データマイニングと従来研究 RelLCM2 について述べる．第 3 節で提案する頻出飽和連言枚挙アルゴリズム ffLCM について説明し，第 4 節でそのアルゴリズムの正当性についてを述べる．更に，第 5 節で予備的実験結果を示し，第 6 節で結論と今後の課題について述べる．

2　準備

2.1　関係データマイニング

関係データマイニング (MRDM) では，ユーザが注目する述語—これを目標述語，あるいはキー (アトム) という—に対応する関係のほか，それに関連する複数の関係表を含むデータベースが与えられている．頻出パターンマイニングは，そのようなデータベースからリテラルの連言で表現されるユーザにとって興味深いパターンを発見する問題である．以下に簡単な例 2.1—文献 [6] の例を単純化したものである —を示す．

例 2.1 (MRDM)．図 1 のデータベース DB_{gf} を考える．

gf
a
b

f
e
d

m
a
b
c

p	
a	b
b	c
c	d
c	e

図 1: 祖父に関するデータベース

DB_{gf} では，それぞれ $gf(X)(f(X),\ m(X))$ で「X が祖父 (女，男) である」，$p(X,Y)$ で「X が Y の親である」という関係を表す．目標述語 (キーアトム) を gf とする．この時，例えば連言 $C = gf(X), m(X), p(X,Y), p(Y,Z)$ は,「祖父である X は男であり，X にはある子供 Y がいて，さらに Y にはある子供 Z がいる」ということを表す．DB_{gf} では，$gf(X)$ を真とする $X = a$, $X = b$ それぞれについて C は真となるので，C は 100%正しいパターンである．　□

本稿では，アトム (あるいはリテラルとも言う) の連言 $a_1, \ldots, a_n$ をデータとして考え，特に連言に変数を含むものをパターン (あるいはクエリ, query) という．

以下では，連言をリスト形式 $[a_1, \ldots, a_n]$ で表すことがある．また，連言 C とアトム p に対して，$[C, p]$ は C の末尾に p を追加した連言を表す．式 E に現れる変数の集合を $Var(E)$，定数の集合を $Const(E)$ と書く．

2.2　飽和連言枚挙アルゴリズム RelLCM2

この節では Garriga らによる飽和連言を枚挙するアルゴリズム RelLCM2 [4] について説明する．

2.2.1 飽和連言

データマイニングにおいて頻出パターンをすべて求める場合，多くのパターンは冗長であるため代表となるパターンだけを求める方が効率がよい．LCM [7, 8] ではそのような代表としてアイテム集合の飽和集合を用いている．RelLCM2 ではアイテム集合を拡張した**飽和連言**という概念を考えている．飽和連言は出現集合を用いて定義される．

定義 1 (出現集合)．連言 C のデータベース DB における出現集合 $covers(C, DB)$ は，次のように定義される：$covers(C, DB) = \{\theta | C\theta \subseteq DB, \theta$は C に出現する変数への代入 $\}$． □

ここでは，データベース DB を一つの連言としてみなしている．なお，DB が明らかな場合はそれを明示せずに単に $covers(C)$ と書く．

飽和連言とは，その出現集合が変化しないという条件の下で，アトムを可能な限り追加してできる連言である．飽和連言は**閉包** (closure) という手続きにより計算される．RelLCM2 では，連言 C の閉包は精密化オペレータで規定されるアトムを C に次々と追加していくことにより計算される．ただし，出現集合の比較を可能にするためには，C に対して**領域制限** *(range-restriced)* [2] ―追加するアトム p 中に含まれる変数や定数は C に全て含まれている―を満たすアトムを生成する精密化オペレータ ρ_{RR} を用いなければならない．RelLCM2 の閉包計算アルゴリズム $\mathrm{closure}_{RR}$ を図 2 に示す．

Algorithm $\mathrm{closure}_{RR}(C)$

Input: 連言：C

Output: 飽和連言：C'

1. $C' \leftarrow C$
2. **repeat**
3. 　　アトム $p \in \rho_{RR}(C')$ を見つける．ただし $covers(C') = covers([C', p])$ かつ $p \notin C'$
4. 　　$C' \leftarrow [C', p]$
5. **until** 該当するアトム p が見つからない
6. **return** C'

図 2: 領域制限閉包アルゴリズム $\mathrm{closure}_{RR}$

この領域制限閉包アルゴリズム $\mathrm{closure}_{RR}$ を用いて，飽和連言は以下のように定義される．

定義 2 (飽和連言)．連言 C は次の条件を満たすとき**飽和** *(closed)* である：$C = \mathrm{closure}_{RR}(C)$． □

2.2.2 prefix 保存拡張

飽和連言 C に対し，アトム $p \notin C$ を追加した $C' = [C, p]$ の領域制限閉包 $\mathrm{closure}_{RR}(C')$ は C とは異なる飽和連言となる．これを C の**飽和拡張**と呼ぶ [7, 8, 4]．この飽和拡張を再帰的に用いることで飽和連言の列挙が可能となる．しかし，この拡張方法をそのまま用いると重複解を生成してしまう．これを解の保持なしに深さ優先的に行う重複回避法として，宇野らは *prefix* **保存飽和拡張** (ppc 拡張、prefix preserveing closure extension) [7, 8] を導入した．Garriga らの RelLCM2 [4] は，その考え方を飽和連言の枚挙に適用したものである．

飽和連言を対象にして ppc 拡張を行うために，対象とするパターン (リテラル) の集合に順序を導入する．RelLCM2 では，まずアトムに順序を導入する： 述語名や定数については例えば辞書順，変数については特別な変数集合 $z_1, z_2, \ldots$ を用意しそのインデックス順とし，各定数は変数より小さいと定め，さらにアトム $p(t_1, \ldots, t_n)$ はリスト $[p, t_1, \ldots, t_n]$ と考える．このアトムの順序を用いて，連言をアトムのリストと考えることにより連言に順序― この順序をここでは**辞書式順序**といい，記号 $<, >$ を用いる ―が導入される．

定義 3 (正規形)．C は連言で $Var(C) = \{x_1, \ldots, x_u\}$ とする．この時，C の**正規形** *(normal form)* $\mathrm{nf}(C)$ とは，次の条件を満たす $C\theta$ である：(i) θ は変数 $x_1, \ldots, x_u$ から $z_1, \ldots, z_u$ への名前替え代入 (renaming substitution) で，(ii) $C\theta$ は上記の順序で最小である． □

定義 4 (コア prefix)．連言 C は正規形であるとする．C の**コア** *prefix (core prefix)* $\mathrm{core}(C)$ とは，$\mathrm{nf}(C)$ の最小 prefix pr で，$covers(pr, DB) = covers(\mathrm{nf}(C), DB)$ を満たすものである． □

定義 5 (prefix 保存飽和拡張)．[4] $C = [q_1, \ldots, q_n]$ を正規形の飽和連言とする．連言 C' が以下の条件を満たす時，C の *prefix* **保存飽和拡張** *(ppc* **拡張***)* であるという：

P1: $C' = \mathrm{nf}(\mathrm{closure}_{RR}([C, p]))$．ただし，$p \in \rho_G(C)$[1]，すなわち，$C'$ は C にアトム $p \notin C$ を加えた $[C, p]$ の領域制限閉包をとり，それを正規形にしたものである．

P2: アトム p はすべての $q \in \mathrm{core}(C)$ について $p > q$ を満たす．[2]

P3: 正規形 C' を得る操作 nf は，C と p に現れるどの変数も変更しない．つまり、正規形を得る操作は $[C, p]$ 中のアトムを並び変える可能性はあるが，変数の名前替えは行わない．

P4: q_j を $q_j < p$ を満たす C の最大のアトムとし，$C[j] = [q_1, \ldots, q_j]$ を C の q_j までの prefix とする．このとき，$[C[j], p]$ は C' の prefix である．これは prefix $C[j]$ が保存され，かつ p と q_j の間で出現する新しいアトムが無いことを意味する． □

図 3 にアルゴリズム RelLCM2 を示す．RelLCM2 は RelLCM2($\mathrm{closure}_{RR}(\emptyset)$) で呼び出される．

図 3 において，C から ppc 拡張によって C' を構成している．以下の説明では，この ppc 拡張の中で，特に C から $[C, p]$ を作る操作 (4 行目) を **se 拡張**と呼び，$[C, p]$ の閉包をとる操作 (7 行目) と区別して呼ぶ場合がある．

3 キー概念を導入した飽和連言枚挙

関係データマイニングでは通常，目標述語 (キー) の概念を導入してパターンをマイニングする．これにより，ユーザが興

[1] 精密化オペレータ $\rho_G(C)$ は，$p \in \rho_{G(C)}$ の条件として $p \notin C$ だけを課す一般的なオペレータである．

[2] C は正規形なので，p が $\mathrm{core}(C)$ の最後のアトム q より大きいことをチェックすればよい．

Algorithm RelLCM2*(C)*

Input: 連言：C

1. **if** C が深さ制約を違反する **then return**
2. **if** C が非頻出 **then return**
3. C を出力する
4. **for** 全ての精密化 $[C,p]$，ただし $p \in \rho_G(C)$ かつ p は core(C) の全てのアトムより大きい
5. **do if** $covers([C,p]) = \emptyset$
6. **then** 精密化をスキップ
7. **else** $C' = \mathrm{nf}(\mathrm{closure}_{RR}([C,p]))$ を求める
8. $C[j] = [q_1, \ldots, q_j]$ ただし、q_j は $q_j < p$ を満たす C で最大のアトム
9. **if** $[C[j],p]$ が C' の prefix
10. **then** RelLCM2(C')
11. **return**

図 3: 頻出飽和連言枚挙アルゴリズム RelLCM2

味のあるパターンにのみに注目することが可能になると同時に，ユーザが興味のある変数に対する代入だけに注目した頻度を定義することができる．しかるに，RelLCM2 ではキーの概念を導入していないため，全ての頻出飽和パターンを枚挙している．さらに，出現集合の定義から，パターンに現れる変数全てに同じように着目した頻度を計算している．

本研究では，キーの概念を導入した頻出飽和連言をマイニングする枠組みを考える．これに基づいて，ユーザが注目しているパターンだけに限定して探索を行う飽和連言枚挙アルゴリズム **ffLCM** (first-order feature-based LCM) について述べる．

3.1 言語バイアスとキー概念

関係データマイニングにおいては，**言語バイアス** (例えば [1]) によりユーザが考慮する対象パターンを規定する．例 2.1 では gf がキーである．この時，例えば連言 $C = gf(X), f(Y), m(Z)$ を考えると，$f(Y)$ や $m(Z)$ に出現する変数は，キーアトム $gf(X)$ の変数 X と無関係である．このような連言は注目しているキーと関連していないので，マイニングの対象とすることは不適切であると考えられる．このことは，以下のように定式化される．

定義 6 (関連リテラル)．[5] キーアトム $key(X)$ にリテラル L が**関連している**—$key(X) \sim L$ と書く—とは，(i) $X \in Var(L)$，あるいは (ii) あるリテラル L_1 が存在して，$key(X) \sim L_1$ かつ $Var(L_1) \cap Var(L) \neq \emptyset$. □

定義 7 (バイアス条件)．マイニングの対象とする連言 C として，以下のバイアス条件を満たすものだけを考える：

- C 中には必ずキーとなるアトムが唯一存在し，C 中の全てのリテラルがキーと関連している．
- C 中に出現する変数はある変数の集合 $\mathcal{V}$ に属する．ただし，$|\mathcal{V}| = \mathcal{V}_{max} \geq 1$.[3]
- C 中に出現する定数はある定数の集合 $\mathcal{C}$ に属する．ただし，$|\mathcal{C}| = \mathcal{C}_{max} \geq 0$. □

キーを考慮した出現集合と飽和連言を次のように定義する．

定義 8 (キー制限出現集合)．C をキー $key(X)$ を含む連言，DB をデータベースとする．この時，**キー制限出現集合** $covers_{key}(C, DB)$ は次のように定義される：[4]

$$covers_{key}(C, DB) = \{\theta_{key} | \theta_{key} \subseteq \theta \in covers(C, DB), Var(\theta_{key}) = Var(key(X))\}.$$

□

定義 9 (キー制限飽和連言)．飽和連言 C は，次のような条件を満たす飽和連言 $C' \supset C$ が存在しない時，**キー制限飽和連言**という：(i) C と C' は同じ領域を持ち (すなわち，$Var(C) = Var(C')$ かつ $Const(C) = Const(C')$)，(ii) C と C' はキー制限出現集合が等しい，すなわち $covers_{key}(C) = covers_{key}(C')$. □

連言 C を含む最小のキー制限飽和連言は一意に定まらないことに注意する (例 3.1 参照)．

キーの概念を導入した場合，頻度はキー制限出現集合により定義される．我々の問題はキー制限飽和連言をすべて枚挙することになるが，キー制限飽和連言を用いて RelLCM2 のような ppc 拡張による探索を行うことを考えると，ppc 拡張の持つ性質 — 任意の飽和連言は，ある飽和連言の ppc 拡張により得られる—が成り立たない．このことを例 3.1 に示す．

例 3.1 (キー制限飽和連言と ppc 拡張による探索)．例 2.1 のデータベース DB_{gf} を対象とし，gf をキーする．また，$\mathcal{V} = \{X, Y\}$ とする．この場合の ppc 拡張による探索の一部を図 4 に示す．

図 4 で，$C = [gf(X), m(X), m(Y)]$，$C_1 = [C, p(X,Y)]$ とすると，$covers_{key}(C, DB_{gf}) = covers_{key}(C_1, DB_{gf})$ で，C_1 はキー制限飽和連言になっている．

ここで $C_2 = [C, p(Y,X)]$ を考えると，$covers_{key}(C, DB_{gf}) \neq covers_{key}(C_2, DB_{gf})$ が成立していて，C_2 もキー制限飽和連言である．しかるに，C_2 はキー制限飽和連言 C_1 から ppc 拡張では—$covers([p(X,Y), p(Y,X)], DB_{gf}) = \emptyset$ のため—得られないことが分かる．

一方，定義 1 の出現集合の場合は C, C_1, C_2 いずれも飽和連言で，C_1, C_2 は C から ppc 拡張により得られている． □

このことから，本研究のキーの概念を導入した頻出飽和連言のマイニングでは，まず定義 1 の出現集合による ppc 拡張を用いた頻出飽和連言の探索を行い，それがキー制限飽和連言であるかの検査を行う方法を考える．

3.2 キーを考慮したリテラル順序

本研究では，バイアス条件 (定義 7) を満たすキーと関連があるリテラルだけからなる飽和連言をマイニング対象としている．

[3] 変数を X_i ($\mathcal{V}_{max} \geq i \geq 1$) と書き，$i$ を変数 X_i のインデックスという．ただし，X_1 はキーに含まれる変数とする．$X_1, X_2, X_3, \ldots$ を $X, Y, Z, \ldots$ と書くこともある．

[4] 対象とする DB が明らかな場合は，$covers$ の場合と同様に，$covers_{key}(C)$ と書く．

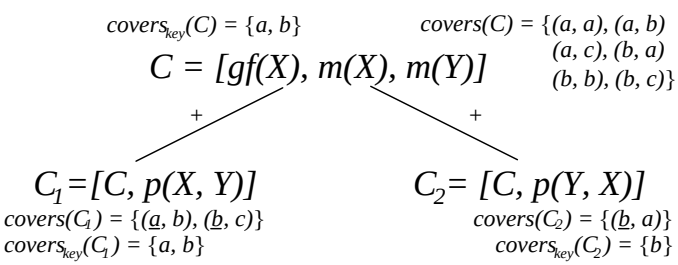

図 4: キー制限出現集合と ppc 拡張による探索：C,C_1,C_2 は飽和連言，C_1,C_2 はキー制限飽和連言．出現集合 *covers* の表記で，代入 $\theta = \{X/t_1, Y/t_2\}$ を (t_1, t_2) と書く．他も同様.

そのため，ppc 拡張を用いた探索過程の途中においても，そのようなバイアス条件を満たすパターンのみ生成する探索を行った方が効率良く探索を行えると考えられる．RelLCM2 の辞書順に基づくリテラル順序では，探索過程でバイアス条件を満たない飽和連言も生成している．

ここでは，バイアス条件を満たすパターンだけを探索過程で生成し，かつマイニングの完全性を失わない—すなわち，バイアス条件を満たすようなすべての頻出飽和連言を枚挙する—リテラル順序を導入する．

定義 10 (キーに関するリテラル順序)**.** $\mathcal{L}$ をバイアス条件 (定義 7) を満たすすべてのリテラル集合とする．$\mathcal{L}$ の分割を以下のように定義する：

$$\begin{aligned}\mathcal{L}_0 &= \{key(X)\}\\ \mathcal{L}_1 &= \{p \mid X \in Var(p), p \neq key(X)\}\\ \mathcal{L}_i &= \{p \mid X_i \in Var(p), \forall X_j \in Var(p) \rightarrow j \geq i\}\\ &\quad (\mathcal{V}_{max} \geq i \geq 2)\end{aligned}$$

ただし，$X_i \in \mathcal{V}$ はインデックス i の変数を表す.

この時，以下のように定義される $\mathcal{L}$ の順序 $\prec$ を**キーに関するリテラル順序**という：(i) 順序 $\prec$ のリテラル集合 $\mathcal{L}_i$ $(\mathcal{V}_{max} \geq i \geq 0)$ への制限は辞書式順序と一致し，(ii) $p \in \mathcal{L}_i$, $q \in \mathcal{L}_j$ で，$i < j$ ならば，$p \prec q$. □

このリテラル順序により $\mathcal{L}$ に全順序 $\prec$ が導入される．$\mathcal{L}$ の各リテラルを昇順に並べてできるリテラル列 $l_0 = key(X), l_1, l_2, \ldots, l_{\mathcal{L}_{max}}$ を考えて，i をリテラル l_i のインデックスという.

キーに関するリテラル順序を用いた場合の正規形，コア prefix の概念も，それぞれ定義 3，定義 4 と同様に定義される．キーに関するリテラル順序を用いると，以下の性質が成立する.

補題 3.1. 連言 C はバイアス条件を満たすとする（すなわち，$\forall l \in C, l \sim key(X)$)．この時，$C$ の存在変数の名前替え C' で，$\mathrm{nf}(C')$ がそのどの prefix もバイアス条件を満たしているものが存在する. □

例 3.2. 連言 $C = key(X), p(X,Z), m(Y), p(Y,Z), m(Z)$ は，バイアス条件を満たすが，その prefix: $key(X), p(X,Z), m(Y)$ はバイアス条件を満たさない $(key(X) \not\sim m(Y)$ であるから)．しかるに，C の名前替え $C' = key(X), p(X,Y), m(Z), p(Z,Y), m(Y)$ は，その正規形 $\mathrm{nf}(C') = key(X), p(X,Y), m(Y), p(Z,Y), m(Z)$ を考えると，そのどの prefix もバイアス条件を満たしている. □

3.3 領域制限閉包の計算

キーに関するリテラル順序を導入すると，領域制限閉包計算は図 5 のように実現できる．ここでは領域制限閉包の計算とともに，prefix 保存のチェックも行っている.

Algorithm closure$_{RR}$−ppc(C, A)

Input: 連言：C, アトム：A

1. **for** $i \leftarrow 1$ **to** $\mathcal{L}_{max}$
2. **do** $p \leftarrow L_i$
3. **if** $p \in C$ **or** p が C に対し領域制限を満たさない **then** 次の p へスキップ
4. $C' \leftarrow [C, p]$
5. **if** $covers(C') = covers(C)$
6. **then if** $p \prec A$ **then return** null
7. **else** $C \leftarrow C'$
8. **return** C

図 5: キーを考慮した領域制限閉包 closure$_{RR}$−ppc

アルゴリズム closure$_{RR}$−ppc(C, A) において，引数 A は se 拡張により連言 C に最後に追加されたアトムである．RelLCM2 では領域制限閉包を求めた後に prefix 保存のチェックを行っていたが (図 3 の 9 行目)，ここではこのチェックを領域制限閉包を求める際に同時に行っている (6 行目)．もし，このチェックで prefix 保存をしていない場合は，アルゴリズムは null を返す.

3.4 頻出飽和連言枚挙アルゴリズム ffLCM

キーを考慮した頻出飽和連言枚挙アルゴリズム ffLCM を図 6 に示す．アルゴリズムは ffLCM$(\emptyset, \emptyset)$ を呼び出すことで動作する．ただし，扱いを統一するため，空の連言 $\emptyset$ は飽和であるとし，そのインデックス $\emptyset.index = -1$ と定める． [5]

9 行目で C_1 — $[C, p]$ の領域制限閉包で ppc 拡張の条件を満たす連言 — がキー制限飽和連言であるかの検査を行う．この検査は，キー制限領域制限閉包の検査手続き (図 7) を is_closure$_{RR,key}(C_1, p)$ で呼び出すことにより行われる．この手続きは二つの場合に分けて考えられる：

1. もし $r \prec p$ なるアトム r が存在し，その領域制限閉包 $C' = \text{closure}_{RR}\text{−ppc}(C_1, r)$ について，

$$covers_{key}(C') = covers_{key}(C_1) \qquad (1)$$

[5] キーアトム $key(X)$ のインデックスが 0 であることに注意．また，空連言 $\emptyset$ に対し，キー $key(X)$ はバイアス条件を満たすとする.

Algorithm ffLCM(C, A)
Input: 飽和連言：C, アトム：A
1. **for** $i \leftarrow A.index + 1$ **to** $\mathcal{L}_{max}$
2. **do** $p \leftarrow l_i$ (l_i :i 番目のリテラル)
3. **if** $p \in C$ **then** 次の p へスキップ
4. **if** p が C に対しバイアス条件を満たさない **then** break
5. $C' \leftarrow [C, p]$ ($*$ se 拡大：C に p を付加 $*$)
6. **if** C' が非頻出
7. **then** 次の p へスキップ
8. **else** $C_1 \leftarrow \text{closure}_{RR}\text{−ppc}(C', p)$
9. **if** $C_1 \neq$ null がキー制限飽和 **then** C_1 を出力
10. **if** $C_1 \neq$ null **then** ffLCM(C_1, p)

図 6: 頻出飽和連言枚挙アルゴリズム ffLCM

が成り立つ場合は，ppc 拡張による探索木においてノード C_1 よりも左側で C_1 を含むキー制限飽和閉包が計算される．従って，C_1 は解として出力しないので false を返す (7 行目).

2. さもなければ，$p \prec r$ なるアトム r について，式 (1) が成り立つ場合は，ノード C_1 の下で C_1 を含むキー制限飽和閉包が計算される．この場合も，C_1 は解として出力しないので false を返す (7 行目).
3. これ以外の場合には，C_1 がキー制限飽和閉包であると判定するので true を返す (8 行目).

Algorithm is_closure$_{RR,key}(C, p)$
Input: 連言：C, アトム：p
1. **for** $i \leftarrow 1$ **to** $\mathcal{L}_{max}$
2. **do** $r \leftarrow l_i$
3. **if** $r \in C$ **or** r が C に関して領域制限を満たしていない
4. **then** 次の r へスキップ
5. $C' \leftarrow \text{closure}_{RR}\text{−ppc}(C, r)$
6. **if** $covers_{key}(C') = covers_{key}(C)$
7. **then return** false
8. **return** true

図 7: キー制限領域制限閉包の検査

アルゴリズム ffLCM(図 6) の 4 行目において，もしリテラル p が連言 C に対してバイアス条件を満たさない—つまり，$[C, p]$ がキーと関連していないか，変数・定数の出現制約を満たしていない—ならば，そこで探索を終了している．これは，キーに関するリテラル順序 (定義 10) により，$p \prec q$ なるどのリテラル q についてもバイアス条件を満たさないという性質による．

8 行目の領域制限閉包計算では，定義 1 の出現集合に基づく閉包 C_1 を求めている．C_1 が ppc 拡張の条件を満足している場合は，ffLCM を再帰的に呼び出している (10 行目)．これにより，頻出飽和連言の探索は深さ優先で行われる．

4 アルゴリズムの正当性について

提案したアルゴリズムの正当性は次の定理による．ここで，連言 C に出現する変数 X で，キーに出現しない変数を C の**存在変数**ということにする．すなわち，$X \in Var(C) \setminus Var(k)$ (ただし k はキーアトム).

定理 4.1 (ffLCM の正当性)**.**

1. アルゴリズム ffLCM で連言 C が出力されるならば、C はバイアス条件を満たす頻出キー制限飽和連言である．
2. C がバイアス条件を満たす頻出キー制限飽和連言ならば、C の存在変数の名前替え C' が存在して，C' はアルゴリズム ffLCM で出力される． □

定理 4.1 の 2 で C の変数の名前替え C' を考えているのは，補題 3.1 による．

このアルゴリズムの正当性の証明は LCM，RelLCM2 の正当性の証明とほぼ同様に行うことができる．ただし, 本研究では, キー制限出現集合 (定義 8), キー制限飽和連言 (定義 9), およびキーに関するリテラル順序 (定義 10) の概念を導入している．キー制限飽和連言についての扱いは，ffLCM アルゴリズム (図 6) の 9 行目でキー制限領域制限閉包の検査手続き (図 7) を行うことに対応している．また、リテラル順序を本論文のように定義しても正当性の証明に影響を与えないことが示される．

5 実験結果

提案したキーに関するリテラル順序順序 $\prec$(定義 10) が辞書式順序と比べて探索にどのような効果があるかを評価するため, 3 つの比較実験を行った．実験結果を表 1〜表 3 に示す.

実験はパターンに出現する変数の数 $\mathcal{V}_{max}$ を変化させて行った．表の属性 (ラベル) で,「キー制限飽和」はキー制限頻出飽和連言数,「ノード数」は探索中に生成された頻出飽和連言数 (探索木の展開ノード数) を表す．ノード数には，一般にキー制限頻出飽和でない頻出飽和連言も含む．また，辞書式順序を用いた場合の括弧内の数字は，バイアス条件を満たす—つまりキーと関連している—連言の数である．また，参考のため,「頻出」として頻出連言の数も示す.

例 2.1 について，(i) パターンに変数だけが出現する場合 (表 1) と，(ii) 変数と定数が出現する場合 (表 2) について生成される連言数を調べた．パターンに出現する定数集合は $\mathcal{C} = \{a, b, c, d, e\}$ である．場合 (i) の展開ノード数に注目すると，変数の数 $\mathcal{V}_{max}$ が増えるに従って，キーに関するリテラル順序を用いることにより，ノード数が減少していることが分かる．$\mathcal{V}_{max} = 4$ の場合で，約 79%ノード数が削減できている．また，場合 (ii) でもほぼ同様の効率化が示されている．

表 3 に，別のデータ *train* を用いた場合の結果を示す．データ *train* は，ある列車がどのような車両を持つかという情報を記述した機械学習で用いられるデータベースである．この場合も，変数の数 $\mathcal{V}_{max}$ が増えるに従って，キーに関するリテラル順序を用いることにより，展開ノード数が最大で約 36%削減されることが分かった．

	辞書式順序						キーに関する順序 $\prec$		
変数	キー制限飽和		ノード数		頻出		キー制限飽和	ノード数	頻出
1	1	(1)	1	(1)	2	(2)	1	1	2
2	4	(3)	5	(3)	14	(10)	3	3	10
3	32	(19)	41	(19)	178	(106)	15	15	82
4	290	(123)	377	(123)	3830	(2254)	78	78	1478

表 1: 例 2.1 でパターンに変数だけが出現する場合の実験結果：括弧内はバイアス条件を満たす連言数

	辞書式順序						キーに関する順序 $\prec$		
変数	キー制限飽和		ノード数		頻出		キー制限飽和	ノード数	頻出
1	5	(5)	5	(5)	10	(10)	5	5	10
2	78	(62)	83	(62)	506	(302)	52	52	198
3	791	(508)	909	(508)	39554	(22898)	323	323	11178
4	7629	(3458)	9161	(3458)		-	1783	1783	-

表 2: 例 2.1 でパターンに変数と定数が出現する場合の実験結果：括弧内はバイアス条件を満たす連言数．-は連言数 $> 4.0 \times 10^5$ を示す．

	辞書式順序						キーに関する順序 $\prec$		
変数	キー制限飽和		ノード数		頻出		キー制限飽和	ノード数	頻出
1	1	(1)	1	(1)	1	(1)	1	1	1
2	2	(2)	2	(2)	2	(2)	2	2	2
3	19	(13)	22	(14)	28	(20)	13	14	20
4	303	(216)	380	(266)	968	(836)	218	248	836

表 3: DB *train* でパターンに変数だけが出現する場合の実験結果：括弧内はバイアス条件を満たす連言数

6 まとめ

本研究では，Garriga らによって提案された RelLCM2 に対して，それにキーの概念を導入した頻出飽和パターン枚挙アルゴリズム ffLCM を提案した．キーの概念を導入し，キーを考慮したキー制限出現集合とキー制限飽和連言の概念を定義した．この枠組みを用いることによって，ppc 拡張による飽和連言の枚挙方法に生じる問題について考察した．次に，キーを考慮した飽和連言を効率よく枚挙するために，リテラル (アトム) 集合に順序関係を導入し，そのキーに関する順序を用いて探索を制御する方法を提案した．このようなキーの概念を導入した頻出飽和連言枚挙アルゴリズム ffLCM を与え，その正当性について述べた．また，キーに関するリテラル順序を用いた場合，キーを考慮しない辞書順のリテラル順序と比較し，探索空間の削減に効果的に働くことを予備的実験によって示した．

今後の課題としては，アルゴリズムの計算量についての解析と，頻出飽和連言枚挙アルゴリズムの効率化の検討がある．また，大規模データに対応するためのデータベース技術を用いた実装方法についても検討する必要がある．

参考文献

[1] Dehaspe, L., Toivonen, H.: Discovery of frequent Datalog patterns. DMKD 1999, 3, pp.7-36 (1999)

[2] De Raedt, L., Ramon, J.: Condensed representations for Inductive Logic Programing. In: 9th Intl, Principles of Knowledge Representation and Reasoning, pp. 438-446 (2004)

[3] Dzeroski, S.: Multi-Relational Data Mining: An Introduction. SIGKDD Explorations Newsletter 2003, Vol.5, Issue 1, pp.1-16 (2003)

[4] Garriga,G. C., Khardon, R., De Raedt, L.: On Mining Closed Sets in Multi-Relational Data. IJCAI 2007, pp.804-809 (2007)

[5] Lavrač, N., Flach, P. A.: An Extended Transformation Approach to Inductive Logic Programming. ACM Trans. Computational Logic, Vol. 2, No. 4, pp.458–494 (2001)

[6] Motoyama,J., Urazawa, S., Nakano, T., Inuzuka, N.: A Mining Algorithm using Property Items Extracted from Sampled Examples. Inductive Logic Programming (ILP2006), LNAI 4455, pp.335–350, 2007.

[7] Uno, T., Asai, T. Uchida, Y., Arimura, H.: An Efficient Algorithm for Enumerating Closed Patterns in Transaction Databases. Discovery Science 2004, LNAI 3245, pp. 16-31 (2004)

[8] 宇野 毅明, 有村 博紀 : データインテンシブコンピューティング —その 2 頻出アイテム集合発見アルゴリズム—. JSAI 2007, Vol.22, No.3, pp.425-436 (2007)

A 補題 3.1 の証明

補題 A.1. 連言 C はバイアス条件を満たすとする（すなわち，$\forall l \in C, l \sim key(X)$）．この時，$C$ の存在変数の名前替え C' で，$\mathrm{nf}(C')$ がそのどの prefix もバイアス条件を満たしているものが存在する． □

証明 1. まず C 内に出現する $X = X_1$ 以外の変数を $\mathcal{V}$ に現れない変数に名前替えする．そのような変数集合を $\tilde{\mathcal{V}}$ とし，名前替えした結果の C を改めて C と書く．この名前替えした結果の C がバイアス条件を満たしていることは明らか．$C = key(X), C^-$ と書く．また，$\mathcal{V}_{used} = \{X\}$ とする．$i = 1$ とする．

(step 1) $C^- = \emptyset$ ならば名前替えの操作終了．

(step 2) さもなければ，$\tilde{\mathcal{L}}_i = \{l \mid X_i \in Var(l), l \in C^-\}$ とし，$C^- := C^- \setminus \tilde{\mathcal{L}}_i$ とする．

(step 3) もし $Var(\tilde{\mathcal{L}}_i) \subseteq \mathcal{V}_{used}$ ならば，$\tilde{\mathcal{L}}_i \subseteq \mathcal{L}_i$. $i := i + 1$ として (step 1) へ．

(step 4) $Var(\tilde{\mathcal{L}}_i) \cap \mathcal{V}_{used} \neq \emptyset$ の場合は，$\tilde{\mathcal{V}}$ 内の変数を含むリテラル $l \in \tilde{\mathcal{L}}_i$ が存在する．

このような l を一つ選択して，l に含まれる $\tilde{\mathcal{V}}$ 内の変数を変数のインデックスが $|\mathcal{V}_{used}|+1$ 以上の変数を順に使って名前替えを行う．この名前替えの操作は $\tilde{\mathcal{L}}_i$ と C^- に適用され，その名前替えした結果を改めてそれぞれ $\tilde{\mathcal{L}}_i$，C^- とする．この時の名前替えに用いた $\tilde{\mathcal{V}}$ 内の変数を $\mathcal{V}_{used}$ に加え，その加えた集合を改めて $\mathcal{V}_{used}$ とする．

(step 5) この (step 4) の操作を $Var(\tilde{\mathcal{L}}_i) \subseteq \mathcal{V}_{used}$ となるまで繰り返す．$i := i+1$ として (step 1) へ．

以上の名前替えの操作の結果得られた各リテラル集合 $\tilde{\mathcal{L}}_i \subseteq \mathcal{L}_i$ $(1 \leq i \leq \mathcal{V}_{max})$ に対して，$\tilde{\mathcal{L}}_i$ を辞書順に並べた連言を C_i とすると，$C' = key(X), C_1, \ldots, C_{\mathcal{V}_{max}}$ が補題の条件を満たす連言になっていることは容易に分かる． □

B　定理 4.1 の証明

定理 B.1 (ffLCM の正当性)**.**

1. アルゴリズム ffLCM で連言 C が出力されるならば、C はバイアス条件を満たす頻出キー制限飽和連言である．
2. C がバイアス条件を満たす頻出キー制限飽和連言ならば、C の存在変数の名前替え C' が存在して，C' はアルゴリズム ffLCM で出力される． □

証明 2. (Sketch)

アルゴリズムの健全性については，ffLCM アルゴリズム (図 6) の 9 行目でキー制限領域制限閉包の検査手続き (図 7) と ppc 拡張の性質から明らかである．

アルゴリズムの完全性について示す．C をバイアス条件を満たす頻出キー制限飽和連言とする．従って，C はバイアス条件を満たす頻出飽和連言である．証明は C の長さに関する帰納法による．補題 3.1 より，C の存在変数の名前替え C' が存在して $\mathrm{nf}(C')$ のどの prefix もバイアス条件を満たすものが存在する．$C' = \mathrm{nf}(\mathrm{closure}_{RR}([C_0, p]))$ と書けるとする．すなわち，C' は頻出飽和連言 C_0 にリテラル p を最後に追加した連言の領域飽和閉包であるとする．この時，C' は ppc 拡張の性質により，$\mathrm{nf}(C_0)$ から ppc 拡張によって得られることが分かる．この時，C_0 はバイアス条件を満たす頻出飽和連言である．よって，帰納法により C' が ppc 拡大により得られることが分かる．

更に，C' がキー制限飽和連言である場合は，その定義よりキー制限領域制限閉包の検査手続き (図 7) の条件を満足するので出力されることは容易に分かる．