To appear in IJCAI 2007

On Mining Closed Sets in Multi-Relational Data

Gemma C. Garriga[*]
Dept. of Computer Science
Uni. Politècnica de Catalunya, Spain
garriga@lsi.upc.edu

Roni Khardon[†]
Dept. of Computer Science
Tufts University, USA
roni@cs.tufts.edu

Luc De Raedt[‡]
Machine Learning Lab
University of Freiburg, Germany
luc.deraedt@cs.kuleuven.be

Abstract

We investigate the problem of mining closed sets in multi-relational databases. Previous work introduced different semantics and associated algorithms for mining closed sets in multi-relational databases. However, insight into the implications of semantic choices and the relationships among them was still lacking. Our investigation shows that the semantic choices are important because they imply different properties, which in turn affect the range of algorithms that can mine for such sets. Of particular interest is the question whether the seminal LCM algorithm by Uno et al. can be upgraded towards multi-relational problems. LCM is attractive since its run time is linear in the number of closed sets and it does not need to store outputs in order to avoid duplicates. We provide a positive answer to this question for some of the semantic choices, and report on experiments that evaluate the scalability and applicability of the upgraded algorithm on benchmark problems.

1 Introduction

The problem of mining frequent itemsets has been a topic of intensive research (see e.g. [Agrawal *et al.*, 1996; Goethals and Zaki, 2004; Han *et al.*, 2000]). Since the number of such sets is huge, it is common and more efficient to restrict the search to *closed* item-sets [Bastide *et al.*, 2000; Zaki, 2004], where a set is closed if all its supersets have strictly lower frequency in the database. The collection of frequent closed sets contains the same information as the overall collection of frequent item-sets, but is much smaller. There is also a growing interest in mining structured data, such as graphs, and more generally multi-relational databases, and the notion of closed sets has also been imported to this richer setup [De Raedt and Ramon, 2004; Yan and Han, 2003]. However, there is no general agreement on an appropriate semantics for closed multi-relational patterns as there exist various possible settings. Some approaches employ θ-subsumption (subgraph homomorphism in the graph setting) to define the frequency of patterns, e.g. [Dehaspe and Toivonen, 2000] but others employ object identity subsumption (subgraph isomorphism) [Malerba and Lisi, 2001; Nijssen and Kok, 2003]. Another variation exist between mining in a single interpretation (graph), or across multiple interpretations. Finally, some authors (e.g. [De Raedt and Ramon, 2004; De Raedt and Dehaspe, 1997]) restrict the implication relation used in defining closures to range-restricted clauses only. In addition to these differences, the notion of a closed set can be coupled with a closure operator that takes a set and calculates its closure and there is more than one way to define such closures. The literature gives the impression that these different choices are unimportant and that algorithmic issues can be studied independently of the semantics. Our investigation shows that this impression is false and that semantics do matter.

Our investigation follows a seminal paper by Uno et al. [2004] who utilized semantic properties in the itemset case to provide a very effective enumeration algorithm for closed sets, named LCM (Linear time Closed pattern Miner). In this paper, we systematically explore the different semantics for mining closed sets in multi-relational databases, and study their effect on the applicability of LCM. Our results identify three types of behaviors that can arise as follows.

First, some definitions specify when a set is closed but make it hard to specify a closure operator. This is the case for the definitions used in CloseGraph [Yan and Han, 2003] and Farmer [Nijssen and Kok, 2003]. As a result this setting is limited to methods that search by applying refinements (in BFS or DFS mode) until a closed set is found.

Secondly, stronger definitions can specify a closure operator yielding unique closures. In this case one can enumerate closed sets by iteratively extending other closed sets (their "parent"), so that the search is more focused. This can be done in the setting of Warmr [De Raedt and Ramon, 2004; Dehaspe and Toivonen, 2000] and Jimi [Maloberti and Suzuki, 2003], which follow normal first order logic semantics. For this setting, we contribute such an enumeration algorithm generalizing LCM. However, just like the simple algorithms, this algorithm must store previously discovered sets to avoid duplicates in the output inducing a serious memory requirement.

Finally, some definitions provide unique closures such that each closed pattern has a *unique* "parent" closed pattern. This

[*]Partly supported by MECD, Spanish goverment, through grant AP2000-4016, and by an EU IHP Marie Curie Fellowship.

[†]Partly supported by NSF Grant IIS-0099446, and by a Research Semester Fellowship Award from Tufts University.

[‡]Partly supported by the EU IST FET project IQ. Now at the Katholieke Universiteit Leuven.

is the case for the Claudien [De Raedt and Dehaspe, 1997] setting, whose semantics is specified using substitutions to variables in a clause. For this case, we provide a full generalization of the LCM algorithm of Uno et al. [2004] that does not need to store previous sets.

Due to space constraints most proofs and some details are omitted from the paper.

2 Problem Definition and Basic Insights

The problems we consider assume that we have a given database DB, a language of patterns $\mathcal{L}$ and a notion of "frequency" measuring how often a pattern occurs in the database. Our databases and pattern are expressed in logical notation. We assume some familiarity with basic notions of logic [Lloyd, 1987] but the following introduces some notation and terminology.

An *atom* is an expression of the form $p(t_1, \ldots, t_n)$ where p is a *relation* of arity n and each t_i is a *term*, i.e. a constant or a variable. This paper employs conjunctions $a_1 \wedge \cdots \wedge a_n$ as data (where terms in the atoms are concrete objects) which we will refer to as *interpretations*. Conjunctions also serve as patterns (where terms can be variables) which are sometimes referred to as *queries*. Notice that when all relations are of arity 0, conjunctions correspond to item-sets. Notice also that if we only have one binary predicate, we can consider it as capturing the edge relation of a graph, so graphs and graph patterns form another special case of relational patterns.

We shall represent conjunctions in list notation, i.e. as $[a_1, \ldots, a_n]$. For a conjunction C and atom p, by $[C, p]$ we refer to the conjunction that results from adding p after the last element of C. For a conjunction C and index i, $C[i]$ is the *prefix* of C, i.e. $[a_1, \ldots, a_i]$. A *substitution* $\theta = \{V_1/t_1, \ldots, V_n/t_n\}$ maps variables to terms. The result of applying a substitution θ to an expression C is the expression $C\theta$ where all variables V_i have been simultaneously replaced by their corresponding term t_i in θ.

2.1 Semantics and Frequent Sets

To relate the data to a query, we employ subsumption. Within inductive logic programming, two notions are popular:

θ-subsumption ([Plotkin, 1970]): C_1 θ-subsumes C_2 if and only if there exists a substitution such that $C_1\theta \subseteq C_2$.

OI-subsumption ([Malerba and Lisi, 2001]): C_1 OI-subsumes C_2 if and only if there exists a substitution $\theta = \{V_1/c_1, \ldots, V_n/c_n\}$ such that $C_1\theta \subseteq C_2$ and the c_i are different terms not occurring in C_1.

We will refer to both cases as subsumption, denoted as $C_1 \preceq C_2$. When we want to identify the substitution witnessing the subsumption we say that $C_1 \preceq C_2$ via substitution θ. Applied to graphs, the former notion corresponds to *subgraph homomorphism*; the latter one to *subgraph isomorphism*.

Two forms of databases will be considered. In the *learning from (multiple) interpretations* (LI) setting, the database contains a set of interpretations. In this case the natural notion of coverage is

$$covers_{LI}(C, DB) = \{D \in DB | C \preceq D\}$$

where each interpretation contributes one element to the cover set.

In the *single interpretation* (SI) setting, the database is a single conjunction and the natural notion of coverage is

$$covers_{SI}(C, DB) = \{\theta | C \preceq DB \text{ via substitution } \theta\}$$

Notice that here the cover set includes substitutions and not interpretations. In both definitions $\preceq$ can be either of the notions of subsumption given above.

We now have four different semantics. The LI-θ setting learns from interpretations using θ-subsumption. This closely corresponds to the setting employed by Warmr [Dehaspe and Toivonen, 2000] and Jimi [Maloberti and Suzuki, 2003]. The LI-OI employs OI-subsumption instead and is employed by CloseGraph [Yan and Han, 2003] and Farmer [Nijssen and Kok, 2003]. Finally, the SI-θ and SI-OI settings learn from a single interpretation only. As we shall see, SI-θ closely corresponds to Claudien's setting [De Raedt and Dehaspe, 1997].

The problem of finding frequent queries is that of finding all queries $C \in \mathcal{L}$ whose frequency $freq(C, DB) \geq t$ for some fixed threshold t. The natural notion for frequency is $freq(C, DB) = |covers(C, DB)|$. It is easy to see that frequency is anti-monotonic for the LI setting, i.e. adding an atom to a conjunction can only decrease the frequency. This important property is used by all frequent set miners.

Remark 1 $freq(C, DB)$ is not anti-monotonic for the SI setting. To see this, consider the query $[q(X)]$ which subsumes $[q(X), p(X, Z)]$. For interpretation $[q(a), p(a, b), p(a, c)]$, $[q(X)]$ has one substitution but $[q(X), p(X, Z)]$ has two substitutions.

Therefore, for the SI setting, we use the notion of relative frequency defined as $relfreq(C, DB) = freq(C, DB)/|D|^v$, where D is the domain (i.e. the set of constants) and v the number of variables appearing in C. Relative frequency is intuitively appealing and it is anti-monotonic for the SI setting.

Remark 2 Note that if we have data from the LI setting we can modify it slightly and interpret it in the SI setting. This is a standard transformation in Inductive Logic Programming. For example consider a dataset with two interpretations $I_1 = [p(a), q(a, b), q(a, c)]$ and $I_2 = [p(d), q(e, f)]$. We add an identifier argument to each predicate and collapse the data into one interpretation: $DB = [p(i_1, a), q(i_1, a, b), q(i_1, a, c), p(i_2, d), q(i_2, e, f)]$. It is easy to see that $freq_{LI}(C, DB) = |covers_{LI}(C, DB)|$ is anti-monotonic in this case. Therefore in this case we have a choice of frequency criterion to use.

2.2 Closures

Finding frequent item-sets is a special case of frequent pattern mining in the LI case. When mining frequent item-sets many patterns are redundant, and therefore one is typically interested in *closed* patterns. Indeed, consider the item-sets $[a, d, e]$, $[a, c, d]$, $[b, c]$ and $[b, d]$. Then $freq([a], DB) = freq([a, d], DB) = 2$. The item a is said to imply d (w.r.t. the database). Therefore, we define:

Definition 3 *A conjunction C implies an atom p in the database DB, denoted $DB \models C \rightarrow [C,p]$, if and only if $covers(C, DB) = covers([C,p], DB)$.*

The rules of the form $C \rightarrow [C,p]$ denote relational association rules that hold with 100 per cent confidence. They are to be understood as implications that are satisfied by all conjunctions in the database. These expressions correspond to the relational association rules, also called query extensions, introduced in Warmr [Dehaspe and Toivonen, 2000]. In case the database consists of a single interpretation only, it does not make sense to allow for atoms p that contain variables that do not appear in C. The reason is that the resulting substitutions are not comparable, cf. also Remark 1. Therefore, in this case one imposes the *range-restriction* requirement [De Raedt and Ramon, 2004], which states that p only contains variables and constants appearing in C. The resulting expression can then be simplified to a (range-restricted) clause $C \rightarrow p$ and corresponds to the pattern type induced by Claudien [De Raedt and Dehaspe, 1997].

In the itemset case the closure of a set of items I is defined as the set of items whose existence in a tuple is implied by the existence of I. Continuing our illustration, one can verify that the closure of $[a]$ w.r.t. the specified tuples is $[a,d]$. An alternative characterization of closed item-sets states that the closure of an item-set I corresponds to the intersection of all tuples in the database subsumed by the item-set I [Zaki, 2004]. Note that intersecting item-sets corresponds to computing the minimally general generalizations of two patterns. This will also play an important role in our work.

The item-set case motivates the iterative closure procedure (ICP) defined below. Note that this procedure employs a *refinement operator* ρ which computes atoms p to be added to the conjunction. Notice also that the condition $DB \models C' \rightarrow p$ can be interpreted according to the four different semantics.

closure(input: pattern $C = q_1, \ldots, q_n$;ρ: ref. operator)
1 $C' \leftarrow C$
2 **repeat**
3 Find an atom $p \in \rho(C')$ s.t. $DB \models C' \rightarrow p$
4 $C' \leftarrow [C', p]$
5 **until** no such atom is found
6 output C'

We first consider a general refinement operator ρ_G that imposes no restrictions on p other than that $p \notin C'$. Some of our theorems below using the normal form require the following syntactic version for the input and output of the refinement operator. We will assume that C' uses variables $x_1, \ldots, x_m$ for some m and that new variables introduced form a contiguous sequences starting with x_{m+1}. This does not change the semantics and does not restrict the form of refinements so we still refer to this case as ρ_G. Until Section 4 we assume that ICP uses ρ_G. The following notion is natural:

Definition 4 (Closed conjunctions of atoms) A conjunction C is closed if $\mathrm{closure}(C) = C$.

The ICP algorithm defines closure in a non-deterministic way that may depend on the order of atom additions. Depending on the semantics, the properties of ICP may vary. In particular, the result may or may not be unique, or the algorithm may not always terminate, in which case the result would not be well defined.

Example 5 Consider using θ-subsumption and the dataset $[R_1(1,2),\ R_1(2,1)]$. Then the pattern $R_1(x_1,x_2)$ implies $[R_1(x_1,x_2), R_1(x_2,x_3)]$ as well as $[R_1(x_1,x_2), R_1(x_2,x_3), R_1(x_3,x_4)]$ and so on. So we can add a chain of any size, the closure is not finite in size, and the procedure may not terminate.

Therefore, under θ-subsumption, it is necessary to restrict the atoms that can be added to the initial conjunction. Section 3 gives a solution that gets around this problem by abstracting the properties of the item-set case from a different perspective. Section 4 gives a different solution for the SI setting.

Before presenting these, we briefly consider the LI-OI setting used in CloseGraph [Yan and Han, 2003] and Farmer [Nijssen and Kok, 2003]. Here the closure procedure is guaranteed to terminate but the results need not be unique:

Example 6 Consider the two conjunctions:

$$[e(1,2,a), e(2,3,b), e(3,4,d), e(4,5,a), e(5,6,c)]$$
$$[e(11,12,a), e(12,13,b), e(12,14,c)]$$

This database represents two edge-labeled graphs, and we consider calculating the closure of $C = e(N_1,N_2,a)$. Then ICP may add $e(N_2,N_3,b)$ to get the closed set $[e(N_1,N_2,a), e(N_2,N_3,b)]$. On the other hand it can add $e(N_2,N_3,c)$ to get the closed set $[e(N_1,N_2,a), e(N_2,N_3,c)]$.

2.3 Normal Form

Most relational frequent pattern mining algorithms employ a normal-form based on an ordering of patterns to avoid investigating the same pattern more than once. We use the following order similar to Nijssen and Kok [2003]. We assume that the predicates and constants are ordered (e.g. alphabetically). In addition, we employ a special set of variables $z_1, z_2, z_3, \ldots$, ordered according to their index, and impose that all constants are smaller than these variables. An order over atoms is then induced by the order over lists composed by the predicate name as first element and its list of arguments following that. An order over conjunctions is induced by the order over lists where the atoms in the conjunction are listed in order.

Definition 7 (Normal form) *The normal form $\mathrm{nf}(C)$ of a conjunction C over variables $x_1, \ldots, x_u$ is $C\theta$ where θ is (1) a one to one substitution from $x_1, \ldots, x_u$ to new variables $z_1, \ldots, z_u$ and (2) $C\theta$ is minimal in the order of conjunctions for substitutions of type (1).*

Clearly the normal form of every conjunction is unique. The normal form also satisfies the following useful properties.

Lemma 8 *Let $C = [q_1, q_2, \ldots, q_n]$ be a conjunction in normal form. (i) For any $i \leq n$, $[q_1, \ldots, q_i]$ is in normal form. (ii) For any $i < n$ and any subset of indices $i < j_1 < j_2 < \ldots < j_k \leq n$, $[q_1, \ldots, q_i]$ is a prefix of the normal form of $[q_1, \ldots, q_i, q_{j_1}, \ldots, q_{j_k}]$.*

3 LI-θ: the lgg Closure

One way to define closures in the item-set case is by intersecting the covered tuples. The equivalent in the LI-θ setting is to apply the well-known lgg operator introduced by Plotkin [1970]. The lgg of s_1, s_2 is a conjunction of atoms s that θ-subsumes both s_1 and s_2 i.e. $s \preceq s_1$, and $s \preceq s_2$, and in addition for any other conjunction s' that θ-subsumes both s_1, s_2 it holds that $s' \preceq s$. Plotkin [1970] proved that the lgg is well defined and provided an efficient algorithm for calculating the lgg of two conjunctions. Note that we do not reduce the lgg. For sets of more than two conjunctions we simply calculate the lgg by iteratively adding one conjunction at a time. The result of this process is unique up to variable renaming regardless of the order of adding the conjunctions. The downside here is that, since the size of the lgg may grow as the product of the sizes of the input conjunctions, the total size may grow exponentially in the number of conjunctions.

Definition 9 (lgg closure) Let C be a conjunction and DB a database, then $\mathrm{closure}_{LGG}(C) = lgg(covers_{LI}(C, DB))$.

We note that a generalization of LCM using the idea of lgg is reported in [Arimura and Uno, 2005]. Their setting however is specialized to ordered tree patterns with a matching relation that corresponds to OI subsumption. Interestingly in this setting the size of the lgg is always small.

We now employ Formal Concept Analysis (FCA) [Ganter and Wille, 1998] to show that the lgg closure produces a Galois connection. We consider the formal context $(2^{DB}, \mathcal{L})$ where $\mathcal{L}$ is the set of possible conjunctions, with the partial orders $\subseteq$ on 2^{DB}, and $\preceq$ on $\mathcal{L}$. We define $\psi(C) = covers_{LI}(C, DB)$ and $\phi(S) = lgg(S)$ where $C \in \mathcal{L}$ and $S \subseteq DB$.

Theorem 10 *The mappings $(\phi, \subseteq)$ and $(\psi, \preceq)$ form a Galois connection. More specifically, (1) $S \subseteq S' \Rightarrow \phi(S') \preceq \phi(S)$; (2) $S \subseteq \psi(\phi(S))$; (3) $C \preceq C' \Rightarrow \psi(C') \subseteq \psi(C)$; and (4) $C \preceq \phi(\psi(C))$.*

Theorem 11 *The compositions $\widehat{\Delta} = \psi \cdot \phi$ and $\Delta = \phi \cdot \psi$ are closure operators. That is, they satisfy monotonicity ($C \preceq C' \Rightarrow \Delta(C) \preceq \Delta(C')$), extensivity ($C \preceq \Delta(C)$), and idempotency ($\Delta(\Delta(C)) = \Delta(C)$).*

Using this result, *closed conjunctions* can be formalized as those coinciding with their closure, that is, $\Delta(C) = C$. The result also establishes that for any set of interpretations S of the database, there is a unique conjunction (up to subsumption equivalence) that is the most specific conjunction satisfied in S. Moreover, the number of closed queries is finite, as is the number of interpretations in our database DB.

We now develop an algorithm RelLCM1 that upgrades the first algorithm in Uno et al. [2004] to the LI-θ case. We need a notion of closure extension that allows us to go directly from one closed conjunction to another.

Definition 12 (Closure extension) A conjunction C' is an extension of a conjunction C if $C' = \mathrm{closure}_{LGG}([C, p])$ for $p \in \rho_G(C)$.

RELLCM1(input: closed pattern C)

```
1  if C is not frequent then return
2  if C is already in closed pattern table then return
3  store C in closed pattern table
4  output C
5  for all refinements p ∈ ρ_G(C)
6          if covers_LI([C,p]) = ∅
7           or covers_LI([C,p]) = covers_LI(C)
8            then skip refinement
9            else Calculate C' = nf(closure_LGG([C,p]))
10                RELLCM1(C')
11 return
```

RelLCM1 stores all previously discovered closed sets in a table and in this way avoids calling the procedure twice on the same set. It uses depth first search. Each closed conjunction is refined in all possible ways and closed. The algorithm checks whether the resulting closed set was already discovered, and if not it is output and further refined to identify more closed conjunctions. The algorithm is started by calling RELLCM1($\mathrm{closure}(\emptyset)$). Clearly every conjunction output by the algorithm is closed. The following theorem guarantees completeness, that is, that every closed conjunction is output by the algorithm.

Theorem 13 *If $C \neq \emptyset$ is a closed conjunction in normal form, then there is a closed conjunction C' and a literal $p \in \rho_G(C')$ such that $C = \mathrm{nf}(\mathrm{closure}_{LGG}([C', p]))$.*

Clearly, in any run of the algorithm the number of calls to RelLCM1 is exactly the number of closed sets. While the number of calls is linear we do discover some patterns more than once. The maximum number of patterns discovered is bounded by the branching factor of the refinement operator times the number of closed sets.

4 SI: Range Restricted Closures

We now consider the SI setting (both under θ- and OI subsumption). As argued in Section 2.2 we need to impose range-restriction when defining the closures. Thus the ICP algorithm employs the operator $\rho_{RR}(C)$ that can only generate atoms containing terms already occurring in C.

Definition 14 (Range restricted closure) $\mathrm{closure}_{RR}(C) = closure(C, \rho_{RR})$.

The next lemma shows that $\mathrm{closure}_{RR}(C)$ is well behaved:

Lemma 15 *For any conjunction C and atoms p, q, if $DB \models C \rightarrow p$ then $DB \models [C, q] \rightarrow p$.*

To get a complete generalization of LCM we need to ensure that each conjunction has a unique parent. Then the algorithm does not need to store the enumerated conjunctions in order to avoid duplicates as in RelLCM1. Therefore, we need a more refined notion of closure extension. The following generalizes the corresponding ideas from [Uno *et al.*, 2004].

Definition 16 (Core prefix) Let C be a conjunction in normal form. The core prefix $\mathrm{core}(C)$ of C is the least prefix pr of $\mathrm{nf}(C)$ such that $covers(pr, DB) = covers(\mathrm{nf}(C), DB)$.

Notice that since we are working in the SI setting, if the covers are the same then pr and C must have the same variables.

Definition 17 (Prefix preserving closure extension) Let $C = [q_1, \ldots, q_n]$ be a closed conjunction in normal form. A conjunction C' is a ppc-extension of C if:
P1: $C' = \mathrm{nf}(\mathrm{closure}_{RR}([C,p]))$ with $p \in \rho_G(C)$, i.e. C' is obtained by adding the atom p to C, taking the closure of $[C,p]$ and normalizing.
P2: The atom p satisfies $p > q$, for all $q \in \mathrm{core}(C)$. Note that because C is in normal form this only requires to check that p is larger than the last atom q in $\mathrm{core}(C)$.
P3: The normal form operation defining C' does not change any of the variables in C and p. That is, the normal form may reorder atoms but may not rename atoms in $[C,p]$.
P4: Let $C[j] = q_1, \ldots, q_j$ be the prefix of C up to q_j, where q_j is the largest atom in C s.t. $q_j < p$. Then, $[C[j], p]$ is a prefix of C'. This means that the core prefix is preserved and that no new atom can appear between p and q_j.

The following statements show that closures and cores are well-behaved, and that ppc-extensions define unique parents.

Lemma 18 *Consider any conjunction C in normal form, any subset $C' \subseteq C \setminus \mathrm{core}(C)$, and any atom $p \notin \mathrm{closure}_{RR}(C)$. (1) $\mathrm{closure}_{RR}(C) \subseteq \mathrm{closure}_{RR}([C,p])$. (2) $\mathrm{closure}_{RR}([C,p]) = \mathrm{closure}_{RR}([\mathrm{core}(C),p]) = \mathrm{closure}_{RR}([\mathrm{core}(C),C',p])$.*

It is worth noting that part (2) of the lemma above is violated by the lgg closure: in some cases $covers_{LI}([C,p]) \neq covers_{LI}([\mathrm{core}(C),p])$ and the closures are different.

Theorem 19 *(1) Let C be a closed conjunction and C' a ppc-extension of C obtained as in condition (P1) in Definition 17. Then $\mathrm{core}(C') = [C[j],p]$ where $C[j]$ is given by condition (P4) in Definition 17 . (2) Let C' be a closed conjunction in normal form. Let $T = \mathrm{core}(C') = [L,p]$ (that is, L is a conjunction and p is the last atom in T). Let $C_1 = \mathrm{closure}_{RR}(L)$ and let $C = \mathrm{nf}(C_1)$. Then C' is a ppc-extension of C and C is the only conjunction for which this holds.*

There is one further caveat to consider: the set of closed patterns can be infinite since adding a variable changes the semantics of the conjunction. Therefore one should employ a "depth bound" on the patterns searched for. The algorithm below is started by calling RELLCM2($\mathrm{closure}(\emptyset)$).

RELLCM2(input: closed pattern $C = q_1, \ldots, q_n$)

```
1  if C violates depth bound then return
2  if C is not frequent then return
3  output C
4  for all refinements [C,p] with p ∈ ρ_G(C)
5        and s.t. p is greater than all atoms in core(C)
6        if covers_SI([C,p]) = ∅
7           then skip refinement
8           else Calculate C' = nf(closure_RR([C,p]))
9                Let C[j] = [q_1, ..., q_j], where
10                  [q_j is largest atom in C with q_j < p]
11               if [C[j],p] is a prefix of C'
12                  then RELLCM2(C')
13 return
```

As in the previous section the number of calls to RelLCM2 is exactly the number of closed sets. Notice that the ppc-extension restriction reduces the branching factor as well. Finally we note that the results of this section hold for both the SI-OI and SI-θ settings.

min freq.	RelLCM2 # closed	RelLCM2 # freq	RelLCM2 CF%	RelLCM2-OI # closed	RelLCM2-OI # freq	RelLCM2-OI CF%
0.9	142 (490 s)	743 (391 s)	80.8	96 (5 s)	483 (302 s)	80.1
0.7	248 (493 s)	1297 (499 s)	80.8	159 (8 s)	747 (392 s)	78.7
0.5	561 (658 s)	3247 (723 s)	82.7	358 (18 s)	1761 (605 s)	79.6
0.3	1206 (992 s)	9743 (1023 s)	87.6	756 (29 s)	3647 (726 s)	79.2
0.1	2243 (1008 s)	-	-	1601 (57 s)	7456 (827 s)	78.5
0.0	3443 (1025 s)	-	-	2808 (97 s)	19224 (922 s)	85.3

Table 1: Effect of decreasing the minimum frequency threshold for NCTRER (with up to 4 variables in patterns and a maximum depth bound of 7).

5 Experimental Evaluation

We now present an empirical evaluation of the RelLCM2 algorithm on two datasets.[1] All experiments were run in a PC Pentium IV EM64T 3.2MHz under Linux 10.0.

The first dataset is the NCTR Estrogen Receptor Binding Database (NCTRER), containing 232 molecules, including bond structures.[2] Since each molecule is a separate structure this is natural for the LI setting so the data can be viewed as $DB = \{d_1, ..., d_n\}$. However as discussed in Remark 2 we can embed it as data for the SI setting suitable for RelLCM2. When doing this we can use $|covers_{LI}(C, DB)|$ as our notion of frequency but still use the proper $covers_{SI}(C, \cup_i d_i)$ for implication and closure calculation.

We first investigate the effect of increasing the frequency threshold for the NCTRER dataset where patterns may contain at most 4 variables and depth bound of 7. Table 1 summarizes the results for RelLCM2 and RelCLM2 under object identity (RelLCM2-OI). The table reports the number of patterns in each case and the associated run time in parentheses. The table also gives results for enumerating all frequent queries (this was done using a level-wise style algorithm). The first observation is that the OI restriction alone substantially decreases the number of frequent and closed sets. This partly explains the speed of systems using this restriction reported in the literature. To see the effect of using closures we calculate the compression factor as $CF\% = (1 - \frac{\#closed}{\#freq}) \times 100$. One can see that in both semantic settings closures further decrease the number of patterns and runtime.

We next investigate the effect of increasing the number of variables for the NCTRER dataset for a fixed threshold of 0.3 and a maximum depth bound of 7. Results are summarized in Table 2. One can see that RelLCM2-OI scales much better than RelLCM2 when increasing the number of variables and that the use of closures significantly reduces the number of patterns and runtime. We also observe that the CF% increases when more variables are considered. As an example, one of the closed structures enumerated which captures active molecules is: $[dsstox\_ids(x), active(x), atom(x, y, \text{`C`}), atom(x, z, \text{`O`}), bond(x, z, y, single), bond(x, w, y, double)]$

[1] A direct comparison with a previous system c-armr [De Raedt and Ramon, 2004] was not possible since it forces the use of syntactic constraints which RelLCM2 forbids so the outputs are different.

[2] http://www.epa.gov/ncct/dsstox/sdf_nctrer.html

Vars	RelLCM2 # closed	RelLCM2 # freq	RelLCM2 CF%	RelLCM2-OI # closed	RelLCM2-OI # freq	RelLCM2-OI CF%
1	3 (0.004 s)	5 (0.004 s)	40.0	3 (0.004 s)	5 (0.004 s)	40.0
2	51 (0.2 s)	109 (0.3 s)	53.2	51 (0.2 s)	109 (0.4 s)	53.2
3	303 (3 s)	1197 (8 s)	74.6	216 (3 s)	711 (10 s)	69.6
4	1206 (992 s)	9743 (1023 s)	87.6	756 (29 s)	3647 (726 s)	79.2
5	-	-	-	2100 (724 s)	-	-
6	-	-	-	3707 (11703 s)	-	-

Table 2: Effect of increasing the number of variables for the NCTRER dataset (with frequency threshold 0.3 and maximum depth bound of 7).

Vars	RelLCM2 # closed	RelLCM2 # freq	RelLCM2 CF%
1	7 (0.024 s)	7 (0.024 s)	0.0
2	28 (8.6 s)	35 (15.16 s)	20.0
3	117 (4610.3 s)	158 (7597.4 s)	25.9

Table 3: Effect of increasing the number of variables on Cora with a relative frequency threshold of 0.1.

The second data set is drawn from Cora, a database of computer science research papers [McCallum *et al.*, 1999]. The resulting collection contains 2706 objects and 12439 links. Here we investigate the number of generated closures when increasing the number of variables. Since this is the true SI setting we use relative frequency. The results for RelLCM2 with threshold 0.01 are reported in Table 3 (the numbers for the OI setting are the same for this small number of variables). We see that in this domain closures are effective both in terms of compression and runtime but the effect is less pronounced than in NCTRER. Among the patterns discovered by RelLCM2, there are closed relational sets such as: $[neural\_netw(x), prob\_methods(y), link\_to(p, x, y)]$, saying that a good proportion of researchers publish papers in both areas.

6 Conclusions

The paper investigated different semantic settings for closed clauses in multi-relational datasets. We have demonstrated that variant definitions from the literature have significant implications for properties of closed sets, and algorithms for discovering them. The paper developed relational variants of the LCM algorithm that are a promising alternative for mining closed conjunctions in datasets. The experiments demonstrated that closed sets significantly compress the number of frequent sets and that the new algorithms scale well and can be used to mine large patterns.

References

[Agrawal *et al.*, 1996] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A.I. Verkamo. Fast discovery of association rules. *Advances in Knowledge Discovery and Data Mining*, pages 307–328, 1996.

[Arimura and Uno, 2005] H. Arimura and T. Uno. An output-polynomial time algorithm for mining frequent closed attribute trees. In *Proc. 15th Conference on Inductive Logic Programming*, volume 3625 of *LNCS*, pages 1–19, 2005.

[Bastide *et al.*, 2000] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L. Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Proc. of 1st Conference on Computational Logic*, volume 1861 of *LNCS*, pages 972–986, 2000.

[De Raedt and Dehaspe, 1997] L. De Raedt and L. Dehaspe. Clausal discovery. *Mach. Learn.*, 26(2-3), 1997.

[De Raedt and Ramon, 2004] L. De Raedt and J. Ramon. Condensed representations for Inductive Logic Programming. In *Proc. of the 9th Intl. Conf. on Principles of Knowledge Representation and Reasoning*, pages 438–446, 2004.

[Dehaspe and Toivonen, 2000] L. Dehaspe and H. Toivonen. Discovery of relational association rules. *Relational Data Mining*, pages 189–208, 2000.

[Ganter and Wille, 1998] B. Ganter and R. Wille. *Formal Concept Analysis. Mathematical Foundations*. Springer, 1998.

[Goethals and Zaki, 2004] B. Goethals and M. Zaki. Advances in frequent itemset mining implementations: report on FIMI'03. *SIGKDD Explor. Newsl.*, 6(1):109–117, 2004.

[Han *et al.*, 2000] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 1–12. ACM Press, 2000.

[Lloyd, 1987] J.W. Lloyd. *Foundations of Logic Programming*. Springer Verlag, 1987.

[Malerba and Lisi, 2001] D. Malerba and F.A. Lisi. Discovering associations between spatial objects: An ILP application. In *Proc. 15th Conference on Inductive Logic Programming*, volume 2157 of *LNCS*, pages 156–163, 2001.

[Maloberti and Suzuki, 2003] J. Maloberti and E. Suzuki. Improving efficiency of frequent query discovery by eliminating non-relevant candidates. In *Discovery Science*, volume 2843 of *LNCS*, pages 220–232, 2003.

[McCallum *et al.*, 1999] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *The 16th Int. Joint Conf. on Artificial Intelligence*, 1999.

[Nijssen and Kok, 2003] S. Nijssen and J.N. Kok. Efficient frequent query discovery in FARMER. In *Proc. of the 7th PKDD*, volume 2838 of *LNCS*, pages 350–362, 2003.

[Plotkin, 1970] G. D. Plotkin. A note on inductive generalization. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, pages 153–163. American Elsevier, 1970.

[Uno *et al.*, 2004] T. Uno, T. Asai, Y. Uchida, and H. Arimura. An efficient algorithm for enumerating closed patterns in transaction databases. In *Discovery Science*, pages 16–31, 2004.

[Yan and Han, 2003] X. Yan and J. Han. CloseGraph: mining closed frequent graph patterns. In *Proc. of the 9th ACM SIGKDD*, pages 286–295. ACM Press, 2003.

[Zaki, 2004] M. Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 4(3):223–248, 2004.