
Originally presented at the 8th ACM Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, China, 02–06 November 2009 (pp. 1409–1412).
New York: ACM.

Available from: http://doi.acm.org/10.1145/1645953.1646132

Copyright © ACM, 2009.
The definitive version was presented at CIKM, 2009.

This is the author’s version of the work. It is posted here by permission of ACM for your personal use. Not for redistribution. If your library has a subscription to these conference proceedings, you may also be able to access the published version via the library catalogue.
ABSTRACT

Tree patterns represent important fragments of XPath. In this paper, we show that some classes of tree patterns exhibit such a property that, given a finite number of tree patterns \( P_1, \ldots, P_n \), there exists another pattern \( P \) (tree pattern or DAG-pattern) such that \( P_1, \ldots, P_n \) are all contained in \( P \), and for any tree pattern \( Q \) belonging to a given class \( C \), \( P_1, \ldots, P_n \) are contained in \( Q \) if and only if \( P \) is contained in \( Q \).

Categories and Subject Descriptors
H.2.4 [Query Processing]: Miscellaneous

General Terms
Theory, Algorithms

Keywords
XPath, Tree Pattern, Containment

1. INTRODUCTION

Tree patterns represent important fragments of XPath. Over the last few years there have been extensive research on tree patterns. In particular, the containment of tree patterns has been investigated in several papers including [3], and some interesting structural properties of tree patterns have been observed, e.g., in [1, 4].

In this paper, we show that some tree patterns exhibit such a property that, given a finite number of compatible tree patterns \( P_1, \ldots, P_n \), there exists another pattern \( P \) (tree pattern or DAG-pattern) such that \( P_1, \ldots, P_n \) are all contained in \( P \), and for any tree pattern \( Q \) belonging to a given class \( C \), \( P_1, \ldots, P_n \) are contained in \( Q \) if and only if \( P \) is contained in \( Q \). We call \( P \) a minimal common container (MCC) of \( P_1, \ldots, P_n \) with respect to \( C \). We provide a method for constructing such MCCs. However, we are not concerned with the complexity of the algorithms, as our focus is on the existence of MCCs. The existence of MCCs has a number of applications. For example, it can be used in finding the maximal contained rewriting of tree pattern queries using views [2] under non-recursive, non-disjunctive DTDs and proving such a maximal contained rewriting can be represented by a single tree pattern, for certain classes of queries and views 1.

2. PRELIMINARIES

Let \( \Sigma \) be an infinite set of tags. An XML tree (Xtree) is a tree with every node labeled with a tag in \( \Sigma \). A tree pattern (TP) is a tree with a unique distinguished node, and with every node labeled with a symbol in \( \Sigma \cup \{\ast\} \) (here \( \ast \) is the wildcard which represents any tag), and every edge labeled with either / or //. The path from the root to the distinguished node is called the distinguished path. In drawing a tree pattern, we will use single and double lines to represent /-edges and // -edges respectively, and use a circle to indicate the distinguished node (see Figure 4). Let \( P \) be a TP. We will use \( \Delta \), and \( \Delta \) to denote the distinguished node and the distinguished path of \( P \) respectively.

Note: the TPs in our discussion correspond to the fragment \( \mathcal{P}^{(\!//\!\text{-}\!\text{edge})} \) defined in [3]. Several subsets of \( \mathcal{P}^{(\!//\!\text{-}\!\text{edge})} \) were studied in [3]. In particular, the subset \( \mathcal{P}^{(\!//\!\text{-}\!\text{edge})} \) contains all TPs that do not have \( \ast \)-nodes. In this paper we are also interested in the subset of all TPs in \( \mathcal{P}^{(\!//\!\text{-}\!\text{edge})} \) such that the root is not labeled \( \ast \), no \( \ast \)-node is incident on a // -edge, and no leaf node is labeled \( \ast \). We denote this subset by \( \mathcal{P}^{(\!//\!\text{-}\!\text{edge})} \).

In what follows, for any tree or rooted directed graph \( T \), we will use \( N(T) \) and \( \tau(T) \) to denote the node set and the root of \( T \) respectively. We will also use \( \text{label}(v) \) to denote the label of node \( v \), and call a node labeled \( \tau \) a \( \tau \)-node. If \( (v_1, v_2) \) is a // -edge (resp. / -edge) in \( T \), we will say \( v_2 \) is a \( /\text{-child} \) (resp. // -child) of \( v_1 \).

A matching of a TP, \( P \), in an Xtree, \( t \), is a mapping \( \delta \) from \( N(P) \) to \( N(t) \) satisfying the following conditions: (1) root-preserving, i.e., \( \delta(\tau(P)) = \tau(t) \), (2) label-preserving, i.e., \( \forall v \in N(P) \), either \( \text{label}(v) = \ast \) or \( \text{label}(v) = \text{label}(\delta(v)) \), and (3) structure-preserving, i.e., for every edge \( (x, y) \in P \), if it is a // -edge, then \( \delta(y) \) is a child of \( \delta(x) \); if it is a / -edge, then \( \delta(y) \) is a descendant of \( \delta(x) \), i.e., there is a path from \( \delta(x) \) to \( \delta(y) \). Each matching \( \delta \) produces a node \( \delta(\Delta(P)) \),

1The details are contained in the full version of this paper.
which is called an answer to the TP. We use $P(t)$ to denote the set of all answers of $P$ over $t$. If $T$ is a set of xtrees, we use $P(T)$ to denote $\bigcup_{t \in T} P(t)$.

Let $P$ and $Q$ be TPs. $P$ is said to be contained in $Q$, denoted $P \subseteq Q$, if for every xtree $t$, $P(t) \subseteq Q(t)$. The equivalence of two TPs is defined as two-way containment as usual.

A Boolean pattern [3] is a tree pattern with distinguished node. Given an xtree $t$ and a Boolean pattern $P$, a matching of $P$ in $t$ is a mapping which is root-preserving, label-preserving, and structure-preserving. Given two Boolean patterns $P_1$ and $P_2$, we say $P_1$ is contained in $P_2$, denoted $P_1 \subseteq P_2$, if whenever $P_1$ has a matching in $t$, for all $P_2$ will also have a matching in $t$. To explicitly distinguish Boolean patterns from non-Boolean patterns, we will use $B(\vdash \vdash \vdash)$ to denote the set of Boolean patterns corresponding to tree patterns in $P(\vdash \vdash \vdash)$, and use $\overline{B(\vdash \vdash \vdash \vdash \vdash)}$ to denote the set of Boolean patterns corresponding to tree patterns in $\overline{P(\vdash \vdash \vdash \vdash \vdash)}$.

3. **MINIMAL COMMON CONTAINER OF TREE PATTERNS**

Throughout this section (and without loss of generality), we assume that the root labels of tree patterns $P_1, \ldots, P_n$ are identical.

3.1 **MCC for Boolean Patterns in $B(\vdash \vdash \vdash)$**

**Definition 3.1:** Let $P_1, \ldots, P_n$ be patterns in $B(\vdash \vdash \vdash)$ with identical root labels. A minimal common container (MCC) of $P_1, \ldots, P_n$ wrt $B(\vdash \vdash \vdash)$, denoted MCC($P_1, \ldots, P_n$), is a pattern $P \in B(\vdash \vdash \vdash)$ such that

1. $P_1, \ldots, P_n \subseteq P$, and
2. for any pattern $Q \in B(\vdash \vdash \vdash)$, if $P_1, \ldots, P_n \subseteq Q$, then $P \subseteq Q$.

Note that if $P$ is the MCC of $P_1, \ldots, P_n$, then for any $Q \in B(\vdash \vdash \vdash)$, $P_1 \cup \cdots \cup P_n \subseteq Q$ if and only if $P \subseteq Q$. Furthermore, all MCCs of $P_1, \ldots, P_n$ are equivalent.

We now show MCC($P_1, \ldots, P_n$) always exists. We will focus on the case $n = 2$ first, and extend it to the general case later.

**Definition 3.2:** (closest matching descendant pair) Two nodes $v_1 \in P_1$ and $v_2 \in P_2$ are said to be a matching pair if $\text{label}(v_1) = \text{label}(v_2)$. We use $[v_1, v_2]$ to denote the matching pair. Let $[v_1, v_2]$ and $[u_1, u_2]$ be matching pairs. If $v_1$ is a descendant of $u_1$ and $v_2$ is a descendant of $u_2$, then we say $[v_1, v_2]$ is a matching descendant pair of $[u_1, u_2]$. If there is no matching descendant pair $[x_1, x_2]$ of $[u_1, u_2]$ such that $[v_1, v_2]$ is also a matching descendant pair of $[x_1, x_2]$, then we say that $[v_1, v_2]$ is a closest matching descendant pair (CMDP) of $[u_1, u_2]$.

Note that every matching pair, except $[rt(P_1), rt(P_2)]$, is a CMDP of one and only one other matching pair.

**Example 3.1:** Consider the Boolean patterns in Figure 1 (a) and (b). In the figure we use subscriptions to distinguish nodes with the same label (e.g., $b_1, b_2$ and $b_3$ represent three nodes labeled with $b$). It can be seen that

- $[a_1, a_2], [b_1, b_2], [b_1, b_3], [c_1, c_3], [c_2, c_3], [d_1, d_2], [e_1, e_2]$

\[\begin{array}{c}
\text{(a) } P_1 \\
\text{(b) } P_2 \\
\text{(c) } \text{MCC}(P_1, P_2)
\end{array}\]

**Figure 1:** Boolean patterns $P_1$, $P_2$ and their MCC. Subscripts are used to distinguish nodes with the same label, e.g., $a_1, a_2$ represent two nodes labeled with $a$.

are all matching pairs wrt $P_1$ and $P_2$, and $[b_1, b_2]$, $[b_1, b_3]$, $[c_1, c_3]$, $[c_2, c_3]$, and $[d_1, d_2]$ are CMDPs of $[a_1, a_2]$. In addition, $[c_1, e_2]$ is a CMDP of $[b_1, b_3]$, but it is not a CMDP of $[a_1, a_2]$.

Given a matching pair $[v_1, v_2]$, we can construct a tree pattern $T(v_1, v_2)$ of height 1 as follows.

1. the root of $T(v_1, v_2)$ is labeled with $\text{label}(v_1)$,
2. for every CMDP $[u_1, u_2]$ of $[v_1, v_2]$, add a child $u$ to the root and label it with $\text{label}(u_1)$. If $u_2$ is a /-child of $v_1$, and $u_1$ is a /-child of $v_2$, then label the edge $(rt(T(v_1, v_2)), u)$ with /; otherwise label it with $\vdash$. For example, for the patterns $P_1$ and $P_2$ in Figure 1, $T(a_1, a_2)$ is the pattern consisting of the root and all of its children in MCC($P_1, P_2$) shown in Figure 1 (c).

Note that every node in $T(v_1, v_2)$ corresponds to a CMDP of $[v_1, v_2]$.

We can now construct a Boolean pattern $P \in B(\vdash \vdash \vdash)$ as follows.

Step 1: Initially, let $P = T(rt(P_1), rt(P_2))$.

Step 2: For every leaf node in $P$, if it corresponds to the matching pair $[u_1, u_2]$, then replace it with $T(u_1, u_2)$ (by merging it with the root of $T(u_1, u_2)$).

Step 3: Repeat Step 2 above until no more nodes can be added to $P$, i.e., every leaf node in $P$ corresponds to a matching pair which has no matching descendant pair.

**Example 3.2:** The pattern constructed for the patterns $P_1$ and $P_2$ in Figure 1 is as shown in Figure 1 (c). It is obtained by replacing the node corresponding to $[b_1, b_3]$ in $T(a_1, a_2)$ with $T(b_1, b_3)$.

Note that there is an 1:1 correspondence between the matching descendant pairs of $[rt(P_1), rt(P_2)]$ and descendants of $rt(P)$. We can prove the following theorem.

**Theorem 3.1:** The Boolean pattern $P$ constructed above is a MCC of $P_1$ and $P_2$ wrt $B(\vdash \vdash \vdash)$.

We can construct a MCC of $P_1, \ldots, P_n$ recursively using the formula $\text{MCC}(P_1, \ldots, P_n) = \text{MCC}(P_n, \text{MCC}(P_1, \ldots, P_{n-1}))$. Thus we have

**Theorem 3.2:** For any finite number of Boolean patterns $P_1, \ldots, P_n \in B(\vdash \vdash \vdash)$ that have identical root labels, there is a pattern $P \in B(\vdash \vdash \vdash)$ which is a MCC of $P_1, \ldots, P_n$, wrt $B(\vdash \vdash \vdash)$.

Note that, in the above theorem, there are no restrictions on the Boolean patterns except that they do not involve *.
3.2 Non-Boolean tree patterns in \( \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \)

We call tree patterns with the same label for the roots and the same label for the distinguished nodes compatible patterns. Suppose \( P_1, \ldots, P_n \in \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \) are compatible tree patterns satisfying the following conditions:

(A) the labels on the distinguished paths follow a fixed order, that is, there are no \( P_i, P_j \) and labels \( x, y \) such that in \( \mathsf{DP}(P_i) \) an \( x \)-node is an ancestor of a \( y \)-node, and in \( \mathsf{DP}(P_j) \) a \( y \)-node is an ancestor of an \( x \)-node;

(B) no label is repeated on the distinguished path of \( P_i \) (for \( i \in [1, n] \)).

With conditions (A) and (B) above, the common labels on \( \mathsf{DP}(P_i) \) and \( \mathsf{DP}(P_j) \) occur in the same order in \( \mathsf{DP}(P_i) \) and \( \mathsf{DP}(P_j) \), and for each common label there is a unique node in \( \mathsf{DP}(P_i) \) with that label (for \( i, j \in [1, n] \)).

Before describing the method to construct the MCC, we need to introduce some notations. Given a pattern \( P \) and a node \( v \) on the distinguished path of \( P \), we use \( \mathsf{Sub}_v(P) \) to denote the full subtree rooted at \( v \) and treat it as a Boolean pattern (by disregarding the distinguished node). We also use \( S_v(P) \) to denote the Boolean pattern corresponding to the subtree obtained from \( \mathsf{Sub}_v(P) \) by removing \( \mathsf{Sub}_v(P) \) if \( u \) is the child of \( v \) on the distinguished path. In other words, if \( v \) and \( u \) are nodes on \( \mathsf{DP}(P) \), and \( u \) is a child of \( v \), then \( S_v(P) = \mathsf{Sub}_v(P) - \mathsf{Sub}_u(P) \). In the special case where \( v = \mathsf{DB}(P) \), \( S_v(P) = \mathsf{Sub}_v(P) \) (See Figure 2).

We can now construct a pattern \( P \) from \( P_1 \) and \( P_2 \) as follows:

1. Find the common labels on \( \mathsf{DP}(P_1) \) and \( \mathsf{DP}(P_2) \). Without loss of generality, we assume they are ordered as \( \tau_1, \ldots, \tau_n \) in \( \mathsf{DP}(P_1) \) and \( \mathsf{DP}(P_2) \). It follows that \( \tau_i = \text{label}(rt(P_i)) = \text{label}(rt(P_2)) \) and \( \tau_n = \text{label}(\mathsf{DN}(P_1)) = \text{label}(\mathsf{DN}(P_2)) \).

2. Construct a temporary distinguished path

\[
x_1/x_2/\ldots/x_{n-1}/x_n
\]

such that \( \text{label}(x_i) = \tau_i \), with \( x_n \) being the distinguished node.

3. Suppose \( v_1, \ldots, v_n \) are the nodes labeled \( \tau_1, \ldots, \tau_n \) respectively in \( \mathsf{DP}(P_1) \), and \( u_1, \ldots, u_n \) are the nodes labeled \( \tau_1, \ldots, \tau_n \) respectively in \( \mathsf{DP}(P_2) \). For \( i \in [1, n-1] \), if \( u_{i+1} \) is a \( - \)-child of \( v_i \), then we change the edge \((x_i, x_{i+1})\) to \( /-\)-edge.

4. For \( i \in [1, n] \), find \( \mathsf{MCC}(\mathsf{Sub}_{v_i}(P_1), \mathsf{Sub}_{u_i}(P_2)) \), and add it under \( x_i \) (i.e., merge \( rt(\mathsf{MCC}(\mathsf{Sub}_{v_i}(P_1), \mathsf{Sub}_{u_i}(P_2))) \) with \( x_i \)).

The final pattern \( P \) obtained is illustrated in Figure 3.

Example 3.3: Consider the patterns in Figure 4 (a) and (b). The pattern constructed using the above approach is shown in Figure 4 (c).

Note that we are not concerned with the complexity of the algorithm or the minimality of the constructed pattern, as our focus is the existence of a MCC.

We can show that \( P \) is a MCC of \( P_3 \) and \( P_4 \) (The definition of MCC for non-Boolean patterns is similar to that for Boolean patterns, thus omitted here).

Theorem 3.3: The above pattern \( P \) is a MCC of \( P_1 \) and \( P_2 \) wrt \( \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \).

The above result can be easily extended to any finite number of compatible tree patterns in \( \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \).

Corollary 3.1: For compatible patterns \( P_1, \ldots, P_n \in \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \) that satisfy conditions (A) and (B), there exists \( P \in \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \) such that (1) \( P_1, \ldots, P_n \subseteq P \), and (2) for any \( Q \in \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \), \( P_1, \ldots, P_n \subseteq Q \) if \( P \subseteq Q \).

One may wonder when the MCC of \( P_1, \ldots, P_n \) is equivalent to \( P_1 \cup \ldots \cup P_n \). In other words, when \( \mathsf{MCC}(P_1, \ldots, P_n) \subseteq P_1 \cup \ldots \cup P_n \). From [4] we know that for tree patterns in \( \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \), \( P \subseteq P_1 \cup \ldots \cup P_n \) iff there is \( i \in [1, n] \) such that \( P \subseteq P_i \). Therefore, we have

Corollary 3.2: For compatible patterns \( P_1, \ldots, P_n \in \mathcal{P}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \), if \( \mathsf{MCC}(P_1, \ldots, P_n) \) exists, then \( \mathsf{MCC}(P_1, \ldots, P_n) = P_1 \cup \ldots \cup P_n \) iff there is \( i \in [1, n] \) such that \( P_j \subseteq P_i \) for all \( j \neq i \).

3.3 Boolean patterns in \( \widehat{\mathcal{B}}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \)

We now consider Boolean tree patterns in \( \widehat{\mathcal{B}}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \), which is the set of Boolean patterns corresponding to tree patterns in \( \widehat{\mathcal{P}}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \). Recall that such tree patterns do not have *-nodes incident on /-edges, or dangling *-nodes (i.e., *-nodes which do not have a non-* descendant), and the root is not labeled *. As usual, we consider the case \( n = 2 \) first, and extend it to the general case later.

Definition 3.3: (label-matching pair and /-child pair)
Let \( P_1, P_2 \) be Boolean patterns in \( \widehat{\mathcal{B}}(\mathcal{I}, \mathcal{S}, \mathcal{G}) \). Let \( v_1 \) and \( v_2 \) be nodes in \( P_1 \) and \( P_2 \) respectively. If \( \text{label}(v_1) = \text{label}(v_2) \neq \)
Figure 5: $P_{5,6}$ is the MCC of Boolean patterns $P_5$ and $P_6$ wrt $\mathcal{B}(\mathcal{LMDP})$.

*, then we call $[v_1, v_2]$ a *label-matching pair*. We call $[v_1, v_2]$ a /-child pair, if $v_1$ (for $i = 1, 2$) is connected to its parent via a /-edge. A pair refers to either a label-matching pair or a /-child pair.

Given a pair $[u_1, u_2]$, if $v_1 \in P_1$ is a descendant of $u_1$, and $v_2 \in P_2$ is a descendant of $u_2$, and $[v_1, v_2]$ is a label-matching pair, then we call $[v_1, v_2]$ a label-matching descendant pair (LMDP) of $[u_1, u_2]$. If there is no label-matching descendant pair $[x_1, x_2]$ of $[u_1, u_2]$ such that $[v_1, v_2]$ is also a LMDP of $[x_1, x_2]$, then we say $[v_1, v_2]$ is a closest LMDP (CLMDP) of $[u_1, u_2]$. If $v_1$ is a /-child of $u_1$, $v_2$ is a /-child of $u_2$, then we say $[v_1, v_2]$ is a /-child pair of $[u_1, u_2]$.

Note that it is possible that $[v_1, v_2]$ is both a label-matching descendant pair of $[u_1, u_2]$, and a /-child pair of $[u_1, u_2]$.

Given a label-matching pair or /-child pair $[v_1, v_2]$, we can construct a Boolean tree pattern $T(v_1, v_2)$ of height 1 as follows.

1. The root of $T(v_1, v_2)$ is labeled with $label(v_1)$ if $label(v_1) = label(v_2) \neq *$, otherwise the root is labeled $*$.
2. For every /-child pair $[u_1, u_2]$ of $[v_1, v_2]$, add a /-child to the root. If $[u_1, u_2]$ is also a label-matching pair, then label the new node with $label(u_1)$. Otherwise, label the new node with $*$.
3. If $[v_1, v_2]$ is a label matching pair, then for every CLMDP $[u_1, u_2]$ of $[v_1, v_2]$, add a /-child labeled with $label(u_1)$ to the root.

We can now construct a Boolean tree pattern $P$ as follows:

1. Initially, let $P = T(rt(P_1), rt(P_2))$.
2. For every leaf node in $P$, if it corresponds to the pair $[u_1, u_2]$, then replace it with $T(u_1, u_2)$ (by merging it with the root of $T(u_1, u_2)$).
3. Repeat the above step until no more nodes can be added to $P$, i.e., the leaf nodes in $P$ corresponds to a descendant pair which has no descendant pairs.
4. Remove dangling *-nodes.

It is straightforward to see that the pattern $P$ is in $\mathcal{B}(\mathcal{LMDP})$.

Example 3.4: The pattern constructed for the Boolean patterns in Figure 5 (a) and (b) is shown in Figure 5 (c).

Theorem 3.4: $P$ is a MCC of $P_1$ and $P_2$ wrt $\mathcal{B}(\mathcal{LMDP})$, i.e., (1) $P_1, P_2 \subseteq P$, and (2) for every Boolean pattern $Q \in \mathcal{B}(\mathcal{LMDP})$, if $P_1, P_2 \subseteq Q$, then $P \subseteq Q$.

Similar to Boolean patterns in $\mathcal{B}(\mathcal{LMDP})$, in the general case, we can construct the MCC of $P_1, \ldots, P_n$ recursively using the fact $\text{MCC}(P_1, \ldots, P_n) = \text{MCC}(\text{MCC}(P_1, \ldots, P_{n-1}), P_n)$, hence proving the following result.

Theorem 3.5: For any finite number of Boolean patterns $P_1, \ldots, P_n \in \mathcal{B}(\mathcal{LMDP})$ that have identical root labels, there is a pattern in $\mathcal{B}(\mathcal{LMDP})$ which is a MCC of $P_1, \ldots, P_n$ wrt $\mathcal{B}(\mathcal{LMDP})$.

3.4 Tree Patterns in $\mathcal{B}(\mathcal{LMDP})$

Let $P_1, \ldots, P_n$ be compatible TP in $\mathcal{B}(\mathcal{LMDP})$ satisfying the two conditions below:

(I) The distinguished path of each pattern does not have repeated labels (except $*$).

(II) The common labels (except $*$) on these paths appear in the same order.

We can show that there is a DAG-pattern $P$ (see the definition below) such that $P_1, \ldots, P_n \subseteq P$, and for any $Q \in \mathcal{B}(\mathcal{LMDP})$, if $P_1, \ldots, P_n \subseteq Q$ then $P \subseteq Q$. We call $P$ a minimal common container (MCC) of $P_1, \ldots, P_n$ wrt $\mathcal{B}(\mathcal{LMDP})$. For example, in Figure 6, DAG-pattern $P_{7,8}$ is a MCC of $P_7$ and $P_8$ wrt $\mathcal{B}(\mathcal{LMDP})$.

A DAG-pattern is a directed graph such that (1) every node is labeled with a tag in $\Sigma$ or wildcard $*$, (2) every edge is labeled // or /, (3) there is a unique root which has incoming degree 0, and there is a unique distinguished node. In a DAG-pattern, there may be several paths from the root to the distinguished node. Each of these paths is called a distinguished path. The definitions of matching, containment, and containment mapping of tree patterns can be extended straightforwardly to DAG-patterns.

Theorem 3.6: For tree patterns $P_1, \ldots, P_n \in \mathcal{B}(\mathcal{LMDP})$ that satisfy conditions (I) and (II), there is a DAG-pattern $P$ with no *-nodes incident on // -edges and no dangling *-nodes, which contains $P_1, \ldots, P_n$. Furthermore, for any $Q \in \mathcal{B}(\mathcal{LMDP})$, $P_1, \ldots, P_n \subseteq Q$ iff there is a CM from $Q$ to $P$.

4. CONCLUSION

We showed the existence of MCCs for some tree patterns in $\mathcal{P}(\mathcal{LMDP})$ and $\mathcal{B}(\mathcal{LMDP})$, and provided a way to construct such a MCC.

Acknowledgement This work is supported by Griffith University New Researcher’s Grant (GUNRG36621).

5. REFERENCES

