Revisiting Redundancy and Minimization in an XPath Fragment*

Benny Kimelfeld and Yehoshua Sagiv
The Selim and Rachel Benin School of Engineering and Computer Science
The Hebrew University of Jerusalem
Edmond J. Safra Campus, Jerusalem 91904, Israel
{bennyk,sagiv}@cs.huji.ac.il

ABSTRACT

Redundancy and minimization of queries are investigated in a well known fragment of XPath that includes child and descendant edges, branches, wildcards, and multiple output nodes. Contrary to a published result, a proposed technique does not guarantee minimality or even non-redundancy, and it is unknown whether a non-redundant query is also minimal. It is shown that for two sub-fragments, non-redundancy and minimality are the same, and can be realized by means of simple (local) tests. The latter property is used to prove that testing non-redundancy is NP-complete.

1. INTRODUCTION

Minimization is a basic technique for query optimization. Usually, queries are minimized by eliminating redundant parts [1–3, 9, 11]. So, important tools for minimization are algorithms for testing containments and equivalences among queries [6, 8, 10, 12]. We explore the notions of redundancy, minimization and the connection between them in a fragment of XPath that has been widely studied [4–6, 13]. This fragment, $\mathsf{XP}^{\{//,[],*\}}$, consists of tree patterns with child and descendant edges, branches, and wildcards. A pattern is *minimal* if it is not larger than any equivalent pattern. A pattern is *redundant* if it is equivalent to one of its proper sub-patterns (subtrees); otherwise, it is *non-redundant.*

A basic question is whether a non-redundant pattern is also minimal (the opposite is clearly true). This property holds in some sub-fragments of $\mathsf{XP}^{\{//,[],*\}}$, namely, those obtained by not allowing either wildcards [1, 9], descendant edges [11] or branches (actually, all the patterns without branches are minimal). The reason is that in these sub-fragments, containment is characterized by the existence of a homomorphism [6] (in the case of branch-free patterns, a simple normalization needs to be applied first [7]). However, in the fragment $\mathsf{XP}^{\{//,[],*\}}$ as a whole, containment may exist even if there is no homomorphism [6]. Hence, testing containment in each of the three sub-fragments is in polynomial time, but in $\mathsf{XP}^{\{//,[],*\}}$ it is coNP-complete [6].

There is a claim [4] that every non-redundant pattern of $\mathsf{XP}^{\{//,[],*\}}$ is also minimal. However, we show that the proof is incorrect. Hence, it is unknown whether non-redundancy and minimality are the same in $\mathsf{XP}^{\{//,[],*\}}$.

The flaw in [4] led them to the conclusion that a pattern P is non-redundant if for all nodes n of P, the sub-pattern rooted at n has no redundant branch that emanates from n. But we give a counterexample to this claim. Hence, the algorithm of [4] (which tests containments among branches) does not guarantee non-redundancy (let alone minimality).

Still, the approach of [4] is interesting because it attempts to characterize non-redundancy in terms of a local property, namely, whether there is a containment between a pair of branches that emanate from the same node. In comparison, the obvious characterization of non-redundancy is a global one, that is, there is no redundant node. To determine whether this characterization holds, we need to test whether every sub-pattern obtained by deleting a node (and its descendants) is not contained in the original pattern.

We pursue this approach and show that for two large sub-fragments, non-redundancy has a local characterization and is the same as minimality. Interestingly, testing containment in each of these sub-fragments is coNP-complete. In one sub-fragment, minimization is realized by eliminating redundancy based on containments among branches that emanate from the same node (as done in [4, 5]). In the second sub-fragment, we reason about redundancy by utilizing new concepts, namely, *weak* and *relative* containment, and *weak redundancy.* These concepts are based on relaxing the usual definition of evaluation (of a pattern against an XML document) by allowing the root of the pattern to be mapped to any node of the XML tree (and not just to the root thereof).

We refer to the two sub-fragments as *normal forms.* We define them in terms of natural syntactic properties and the notion of *stability.* Although testing stability is NP-hard, we give simple sufficient conditions indicating that the normal forms cover many of the patterns that are used in practice.

For patterns in the normal forms, minimality can often be proved by very simple arguments. As an example, in the proof of [6] that containment is coNP-hard, the patterns are in both normal forms. Thus, we easily show that testing either non-redundancy or minimality is NP-hard.

In [5], they study the sub-fragment comprising all Boolean patterns B, such that every wildcard node of B has at most one child. They show that their algorithm [4] actually min-

*This research was supported by The Israel Science Foundation (Grant 893/05).

EDBT'08, March 25–30, 2008, Nantes, France.

imizes every pattern of this sub-fragment. Their result is a straightforward corollary of our work, because every pattern of their sub-fragment can be easily transformed into an equivalent pattern in the first normal form.

Our results are firstly given for Boolean patterns. But as in [6], we also consider non-Boolean patterns with an arbitrary number of output nodes. We correct a minor flaw in the reduction of [6] by showing how to transform a general pattern into its *b-version,* which is Boolean. We prove that a pattern is redundant if and only if so is its b-version. Consequently, our results carry over to general patterns.

2. FORMAL SETTING

2.1 Trees and XML

In this work, a tree t has a designated node, called the *root* and denoted by $root(t)$, such that every other node of t is reachable from $root(t)$ through a unique directed path. In a *labeled* tree, every node n has a *label* which is denoted by $label(n)$. The sets of nodes and edges of a tree t are denoted by $\mathcal{N}(t)$ and $\mathcal{E}(t)$, respectively. In figures, the direction of edges is not explicitly shown, but is assumed to be from top to bottom.

Consider a tree t. If there is an edge $(n_1, n_2) \in \mathcal{E}(t)$, then node n_1 is the *parent* of n_2, while n_2 is a *child* of n_1. If t has a directed path from n_1 to n_2, then n_1 is an *ancestor* of n_2 (and n_2 is a *descendant* of n_1). The node n_1 is a *proper ancestor* of n_2 (and n_2 is a *proper descendant* of n_1) if, in addition, $n_1 \neq n_2$.

If a tree t' satisfies $\mathcal{N}(t') \subseteq \mathcal{N}(t)$ and $\mathcal{E}(t') \subseteq \mathcal{E}(t)$, then it is a *subtree* of the tree t. Given a node n of t, we denote by t^n_Δ the subtree of t that is rooted at n and induced by all the descendants of n. The subtree $t - n$ is obtained from t by pruning t^n_Δ. Let $r = root(t)$ and n be a child of r. A *branch* of t consists of the edge (r, n) and the subtree t^n_Δ, and is denoted by $t^r_{n\Delta}$. If the root of t has exactly one child, then we say that t itself is a *branch.*

A *leaf* of a tree t is a node without outgoing edges. The set of leaves of t is denoted by $leaves(t)$. The *height* of t, denoted $height(t)$, is the maximal number of edges in a path from the root to a leaf.

We consider two types of labeled trees that represent XML documents and queries, respectively. A document is called an *XML tree* (or *tree* for short) and its labels come from an infinite set Σ. By $\mathbf{T}_\Sigma$ we denote the set of all the trees with labels from Σ.

2.2 Patterns

Queries are called *patterns.* They are different from XML trees in three aspects. First, the labels of a pattern come from the set $\Sigma \cup \{*\}$, where $*$ is the "wildcard" symbol ($* \notin \Sigma$). Second, a pattern P has two types of edges: $\mathcal{E}_/(P)$ is the set of *child edges* and $\mathcal{E}_{//}(P)$ is the set of *descendant edges.* Third, a pattern P has a k-tuple ($k \geq 0$) of *output* nodes that is denoted by $out(P)$ (i.e., $out(P) \in \mathcal{N}(t)^k$). If $k = 0$ then the pattern is *Boolean.* For example, Figures 6 and 7 show patterns. Nodes are circles with labels inside them. Child edges and descendant edges are depicted by single and double lines, respectively. Output nodes are denoted by thicker circles. If there are multiple output nodes, then we write next to each one its index in the output tuple. (In our examples, each k-tuple consists of distinct nodes, but this is not required in general).

Note that patterns represent an extension of the fragment $\mathsf{XP}^{\{//,[\,],*\}}$ of XPath that was investigated in [4,6,13] and is described by the grammar

$$q \Longrightarrow\ q/q \ \mid\ q//q \ \mid\ q[q] \ \mid\ l \ \mid\ *$$

where l denotes a label in Σ.

A pattern is *linear* if it forms a path; that is, each node has at most one child. $\Sigma[P]$ denotes the set of labels of Σ that appear in P. Note that the wildcard $*$ may appear in P, but is not in $\Sigma[P]$.

Let P be a pattern and n be a new node (i.e., $n \notin \mathcal{N}(P)$). The patterns n/P and $n//P$ are obtained by creating a child edge and a descendant edge, respectively, that connect n to $root(P)$. If n is labeled with $*$, then we write $*/P$ and $*//P$.

2.3 Applying Patterns to Trees

We now define the result of applying a pattern to a tree.

DEFINITION 2.1. *An* embedding *of a pattern P in a tree t is a mapping $e : \mathcal{N}(P) \to \mathcal{N}(t)$ with the following properties.*

1. Root preserving. $e(root(P)) = root(t)$.
2. Label preserving. *For all nodes $n \in \mathcal{N}(P)$, either $label(n) = *$ or $label(n) = label(e(n))$.*
3. Child preserving. *For all edges $(n_1, n_2) \in \mathcal{E}_/(P)$, node $e(n_2)$ of t is a child of $e(n_1)$.*
4. Descendant preserving. *For all edges $(n_1, n_2) \in \mathcal{E}_{//}(P)$, node $e(n_2)$ of t is a proper descendant of $e(n_1)$.*

Consider a pattern P and let $out(P) = (m_1, \ldots, m_k)$. An embedding e of P in a tree t *produces* $(e(m_1), \ldots, e(m_k))$, which is a k-tuple of nodes of t. The *result* of applying P to the tree t, denoted by $P(t)$, is the set of tuples produced by all embeddings from P to t.

For a *Boolean* pattern B, the result $B(t)$ is either $\{()\}$ or $\emptyset$, depending on whether there is an embedding of B in t. Note that $\{()\}$ is interpreted as **true** while $\emptyset$ stands for **false**. Thus, B is a Boolean formula that expresses whether there is an embedding in the given tree.

For clarity of presentation, when we specifically refer to a Boolean pattern, we call it a *b-pattern* and denote it by B. When we just say "pattern," it could be either Boolean or non-Boolean.

Recall that an embedding is root preserving. Now, we consider mappings that do not satisfy this property. Consider a pattern P and a tree t. A mapping $e : \mathcal{N}(P) \to \mathcal{N}(t)$ that satisfies the last three properties of Definition 2.1 (but not necessarily the first) is a *weak embedding.* That is, a weak embedding preserves labels as well as child and descendant edges. $P^w(t)$ denotes the set of tuples produced by all the weak embeddings of P in t. Note that $P(t) \subseteq P^w(t)$. In particular, if B is a b-pattern, then $B^w(t)$ is a Boolean value that specifies whether there is a weak embedding of B in t.

2.3.1 Models

A tree t is a *model* of a b-pattern B if $B(t) =$ **true**. Reasoning about b-patterns is made easier by considering only canonical models [6] that are defined as follows.

We assume that $\perp$ is a special label of Σ that does not appear in any of the patterns that we use.

A *canonical* model for a b-pattern B is any tree t that is obtained from B by applying the following two steps. First,

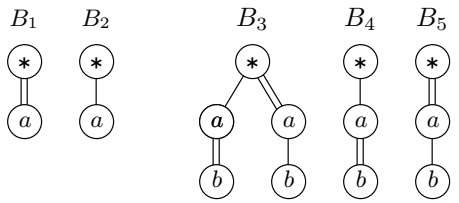

Figure 1: Equivalence vs. weak equivalence

each occurrence of the label $*$ is replaced with $\bot$. Second, each descendant edge is replaced with a path of one or more edges, where all the internal nodes are labeled with $\bot$.

As an example, Figure 2 shows a b-pattern $B_{\min}$ and a canonical model t thereof.

We use $Mod(B)$ and $CMod(B)$ to denote the set of all models and all canonical models of B, respectively.

2.4 Containment and Equivalence

The following definition is just the usual one.

DEFINITION 2.2 (Containment/Equivalence). *Consider two patterns P_1 and P_2 that have the same number of output nodes. If for all trees $t \in \mathbf{T}_\Sigma$, it holds that $P_1(t) \subseteq P_2(t)$, then P_1 is* contained *in P_2, denoted by $P_1 \sqsubseteq P_2$.*

P_1 and P_2 are equivalent*, denoted by $P_1 \equiv P_2$, if $P_1 \sqsubseteq P_2$ and $P_2 \sqsubseteq P_1$, that is, $P_1(t) = P_2(t)$ for all trees $t \in \mathbf{T}_\Sigma$.*

The next definition is based on weak embeddings.

DEFINITION 2.3 (Weak Containment and Equivalence). *P_1 is* weakly contained *in P_2, denoted by $P_1 \sqsubseteq^w P_2$, if for all trees $t \in \mathbf{T}_\Sigma$, it holds that $P_1^w(t) \subseteq P_2^w(t)$.*

P_1 and P_2 are weakly equivalent*, denoted by $P_1 \equiv^w P_2$, if $P_1 \sqsubseteq^w P_2$ and $P_2 \sqsubseteq^w P_1$, namely, $\forall t \in \mathbf{T}_\Sigma(P_1^w(t) = P_2^w(t))$.*

Clearly, if P_1 and P_2 are equivalent, then they are also weakly equivalent. However, the opposite direction does not always hold, as the following example shows.

EXAMPLE 2.4. *Consider the b-patterns $B_1, \ldots, B_5$ of Figure 1. Clearly, B_1 and B_2 are only weakly equivalent (but not equivalent). B_3 is contained in both B_4 and B_5 (because B_4 and B_5 are isomorphic to the left and right branches, respectively, of B_3). But neither B_4 nor B_5 is contained in B_3. In fact, B_4 is not even weakly contained in B_3 (because only B_3 requires that there are a parent and a child labeled with a and b, respectively). However, B_5 is weakly contained in B_3, and therefore, B_3 and B_5 are weakly equivalent.*

Consider two b-patterns B_1 and B_2. To determine whether there is a containment (or equivalence) between them, it is enough to consider their canonical models [6]. Formally, $B_1 \sqsubseteq B_2$ if and only if $CMod(B_1) \subseteq Mod(B_2)$ (that is, $B_2(t) = \mathbf{true}$ for all $t \in CMod(B_1)$). The following proposition shows a similar result for weak containment.

PROPOSITION 2.5. *For all b-patterns B_1 and B_2, we have: $B_1 \sqsubseteq^w B_2$ if and only if $B_2^w(t) = \mathbf{true}$ for all $t \in CMod(B_1)$.*

2.4.1 Isomorphism and G-Isomorphism

Containment and equivalence are semantic properties of patterns. Next, we consider syntactic properties.

Two patterns are *isomorphic* if they are identical up to renaming of nodes. Formally, P_1 and P_2 are isomorphic if there is a bijection $\varphi : \mathcal{N}(P_1) \to \mathcal{N}(P_2)$ that preserves labels, child edges, descendant edges and output nodes (that is, $label(n) = label(\varphi(n))$, if (n_1, n_2) is a child edge then so is $(\varphi(n_1), \varphi(n_2))$, etc.). Obviously, the existence of an isomorphism φ is sufficient but not necessary for $P_1 \equiv P_2$.

A weaker notion is *graph isomorphism* (*g-isomorphism* for short) which is a bijection $\psi : \mathcal{N}(P_1) \to \mathcal{N}(P_2)$ that only preserves the edges; that is, if (n_1, n_2) is an edge of P_1, then $(\varphi(n_1), \varphi(n_2))$ is an edge of P_2 (but the two edges may be of different types). A g-isomorphism is neither necessary nor sufficient for equivalence, but when it exists, it means that the two patterns have the same graph structure and, in particular, the same size.

2.5 Basic Observations

Next, we describe some basic observations that are implied by weak containment and weak equivalence (and, hence, also by containment and equivalence). The first observation shows a connection between height and (weak) containment.

PROPOSITION 2.6. *If the patterns P and P' satisfy $P \sqsubseteq^w P'$, then $height(P) \geq height(P')$. Therefore, if $P \equiv^w P'$, then $height(P) = height(P')$.*

Weak equivalence and weak containment imply some relationships between the labels of the two patterns.

PROPOSITION 2.7. *If the patterns P and P' satisfy $P \equiv^w P'$, then $label(root(P)) = label(root(P'))$.*

PROPOSITION 2.8. *If the patterns P and P' satisfy $P \sqsubseteq^w P'$, then $\Sigma[P] \supseteq \Sigma[P']$. Therefore, if $P \equiv^w P'$, then $\Sigma[P] = \Sigma[P']$.*

Weak equivalence of b-patterns can be reduced to equivalence of Boolean branches (which have roots with exactly one child). The converse also holds if a descendant edge connects the root of each branch to its child.

PROPOSITION 2.9. *Consider the b-patterns B and B'. If the nodes n and n' have the same label and are new (namely, $n_1, n_2 \notin \mathcal{N}(B) \cup \mathcal{N}(B')$), then the following holds. $B \equiv^w B'$ if and only if $n//B \equiv n'//B'$.*

Weak containment (and weak equivalence) between two Boolean branches implies the same between the b-patterns obtained by removing the roots.

PROPOSITION 2.10. *Let r and r' be the roots of the b-patterns B and B', respectively. If n and n' are children of r and r', respectively, and $B^r_{n\,\Delta} \sqsubseteq^w B'^{r'}_{n'\,\Delta}$, then $B^n_\Delta \sqsubseteq^w B'^{n'}_\Delta$. Therefore, if $B^r_{n\,\Delta} \equiv^w B'^{r'}_{n'\,\Delta}$, then $B^n_\Delta \equiv^w B'^{n'}_\Delta$.*

Finally, the next proposition is a direct corollary of Propositions 2.7, 2.9. and 2.10,

PROPOSITION 2.11. *Let n_1 and n_2 be new nodes that do not appear in the b-patterns B_1 and B_2. If $n_1//B_1 \equiv^w n_2/B_2$, then $n_1//B_1 \equiv n_2//B_2$.*

3. REDUNDANCY AND MINIMIZATION

In this section, we discuss the notions of redundancy and minimization of patterns.

3.1 Redundancy of Patterns

We define two notions of redundancy. The first definition says that a pattern is redundant if it has an equivalent proper sub-pattern.

Definition 3.1 (Redundant Pattern). *A pattern P is* redundant *if P has a proper subtree P', such that $P' \equiv P$; otherwise, P is* non-redundant.

The second definition says that a node of a pattern is redundant if that node (and its descendants) can be removed while preserving equivalence.

Definition 3.2 (Redundant Node). *A node n of a pattern P is* redundant *in P if $P - n \equiv P$; otherwise, n is* non-redundant *in P.*

Note that if the pattern P' is a subtree of P, then $out(P')$ is obtained from $out(P)$ by removing the nodes that are not in P'. In particular, P and P' are comparable only if P' includes all the output nodes of P. Consequently, an output node is never redundant.

The next proposition shows that existence of a redundant leaf is necessary and sufficient for redundancy in a pattern.

Proposition 3.3. *A pattern is redundant if and only if it has a redundant leaf.*

Proof. The "if" direction is clear from the definitions. For the "only if" direction, it suffices to show the following. If $P' \equiv P$ for some proper subtree P' of P, then there is a leaf n, such that $P - n \sqsubseteq P$ (note that $P \sqsubseteq P - n$ is obvious). By Proposition 2.6, P' has the same height as P, and hence, P' contains the root of P. Since P' is a proper subtree of P, there is a leaf n of P that is not in P'. Therefore, $P - n \sqsubseteq P' \equiv P$. □

Following is a variant of Definitions 3.1 and 3.2 that uses weak equivalence instead of equivalence.

Definition 3.4 (W-Redundancy). *A pattern P is* w-redundant *if P has a proper subtree P', such that $P' \equiv^w P$. A node n of P is* w-redundant *in P if $P - n \equiv^w P$.*

The proof of the following proposition is similar to that of Proposition 3.3

Proposition 3.5. *A pattern is w-redundant if and only if it has a w-redundant leaf.*

Since equivalence implies weak equivalence, it follows that if a pattern is redundant then it is also w-redundant. Consequently, if a pattern is not w-redundant then it is also non-redundant; that is, non-w-redundancy is stronger than non-redundancy (it is actually strictly stronger as the next example shows).

As an example, consider the b-pattern B_3 of Figure 1. B_3 is non-redundant; however, it is actually w-redundant. (Recall that Example 2.4 shows that B_3 is weakly equivalent to B_5, which is the same as the right branch of B_3.)

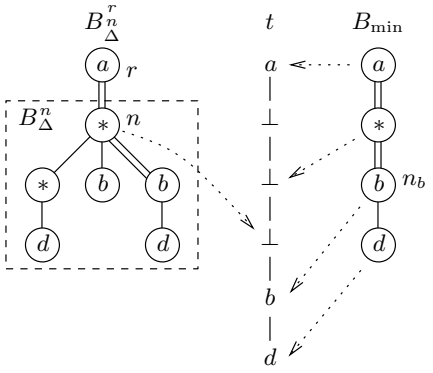

Figure 2: A counterexample to past claims

3.2 Minimization of Patterns

For a pattern P, we use size(P) to denote the number of nodes of P, i.e., $|\mathcal{N}(P)|$.

The problem of *minimizing* a pattern P is to find an equivalent pattern P_m (i.e., $P_m \equiv P$) that has the minimum size. We use $\min_{\equiv}(P)$ to denote the size of the smallest pattern that is equivalent to P. If $\text{size}(P_m) = \min_{\equiv}(P_m)$, then P_m is *minimal*. Similarly, $\min^w_{\equiv}(P)$ denotes the minimum size of a pattern P', such that $P' \equiv^w P$. A pattern P_m is *w-minimal* if its size is $\min^w_{\equiv}(P_m)$. Clearly, $\min^w_{\equiv}(P) \leq \min_{\equiv}(P)$.

3.3 Eliminating Redundancy and Minimizing

By definition, a minimal pattern is non-redundant. However, whether the converse also holds is an open problem. In [4], it was claimed that a non-redundant pattern is minimal. In this section, we show that their proof is wrong. This section and the next one consider only b-patterns.

In our terminology, the flawed argument[1] of [4] is phrased as follows. If B is a b-pattern with a root r and $B^r_{n\,\Delta}$ is a branch of B, then

$$\min_{\equiv}(B^r_{n\,\Delta}) = 1 + \min_{\equiv}(B^n_\Delta)\,.$$

The above equation is equivalent to claiming that the branch $B^r_{n\,\Delta}$ is minimal if and only if the b-pattern B^n_Δ is minimal.

The above equation is wrong because $B^r_{n\,\Delta}$ may be redundant (and, hence, larger than the minimum) even if B^n_Δ is minimal. This is shown in the following example.

Example 3.6. *Consider the b-pattern $B^r_{n\,\Delta}$ that is shown in Figure 2. The following argument proves that B^n_Δ is non-redundant, and by the results of Section 4, it is also minimal. If we remove any leaf m of B^n_Δ and construct a canonical model t (for $B^n_\Delta - m$) by replacing the descendant edge with a path of length 2, then $B^n_\Delta(t) =$ **false**. $B^r_{n\,\Delta}$, however, is redundant because it is equivalent to $B_{\min}$ (which is also shown in Figure 2). Note that $B_{\min}$ is the same as the path from the root of $B^r_{n\,\Delta}$ to the rightmost leaf, and therefore, $B^r_{n\,\Delta} \sqsubseteq B_{\min}$. The other direction, $B_{\min} \sqsubseteq B^r_{n\,\Delta}$, is proved by observing that if there is an embedding $e : B_{\min} \to t$, then an embedding of $B^r_{n\,\Delta}$ in t can be obtained by mapping n to the*

[1]This argument is stated in the last sentence on the fourth page of [4] (and no proof is given).

parent of $e(n_b)$, as illustrated in the figure. Consequently, $min_{\equiv}(B^{r}_{n\,\Delta}) = 4$, whereas $1 + min_{\equiv}(B^{n}_{\Delta}) = 7$.

$B^{r}_{n\,\Delta}$ in the above counterexample is of the form $\hat{n}//\hat{B}$. When minimizing a b-pattern of this form, it is not enough to replace $\hat{B}$ with a minimal b-pattern that is equivalent to $\hat{B}$ (as claimed in [4]). Rather, we need to replace $\hat{B}$ with a minimum-size b-pattern that is weakly equivalent to $\hat{B}$ (by Proposition 2.9, this replacement preserves equivalence). In other words, the following equality is correct for patterns of the form $\hat{n}//\hat{B}$ (the formal proof is based on Propositions 2.9 and 2.10, and Lemma 1 of [4]).

$$\min_{\equiv}(\hat{n}//\hat{B}) = 1 + \min^{w}_{\equiv}(\hat{B})$$

The flawed argument led [4] to propose the following incorrect algorithm for minimizing a b-pattern B. Traverse the nodes of B top down. When visiting a node n, test whether n has a child m that is redundant in B^{n}_{Δ} and if so, prune B^{m}_{Δ} from B. Note that this algorithm uses an oracle for testing containment (which is a coNP-complete problem [6]). It is unknown whether minimization can be achieved by means of redundancy elimination. But the algorithm of [4] is actually wrong because it fails to eliminate redundancy in some cases. For example, it does not remove any part of the redundant b-pattern $B^{r}_{n\,\Delta}$ of Figure 2 (and Example 3.6). Note that a correct algorithm for redundancy elimination is the one implied by Proposition 3.3. Namely, while there is a leaf n, such that $B - n \sqsubseteq B$, remove n from B.

3.4 Redundant Branches in B-Patterns

The rationale for the top-down algorithm of [4] is to improve efficiency by testing containments among branches, rather than between the original query and another one that is almost as large. This approach is based on removing redundant branches that are defined as follows.

Definition 3.7 (Redundant Branch). *Suppose that B is a b-pattern with a root r. If a child n of r is redundant in B (see Definition 3.2), then the branch $B^{r}_{n\,\Delta}$ is* redundant.

The next proposition follows from Lemma 1 of [4].

Proposition 3.8. *A branch R of a b-pattern B is redundant if and only if B has a branch $R' \neq R$, such that $R' \sqsubseteq R$.*

The following is analogous to Definition 3.7.

Definition 3.9 (W-Redundant Branch). *Let B be a b-pattern with a root r. If a child n of r is w-redundant in B (see Definition 3.4), then the branch $B^{r}_{n\,\Delta}$ is* w-redundant.

Example 3.10. *Consider the b-pattern B_6 of Figure 3. The bottom of this figure depicts the branches of B_6 that are denoted by $R_1, \ldots, R_4$. The branch R_1 is redundant (and, hence, w-redundant) because $R_3 \sqsubseteq R_1$. It is more difficult to reason about w-redundancy of branches. For example, consider the b-pattern B_7 and its branches R_5, R_6 and R_7 of Figure 3. The branch R_5 is not w-redundant (and, therefore, not redundant). To see that, let B'_7 be the result of removing R_5 from B_7. It is easy to verify that there is no weak embedding of B_7 in the canonical model of B'_7 that is obtained by replacing each descendant edge with two edges. Note that R_5 is not w-redundant, even though $R_6 \sqsubseteq^{w} R_5$.*

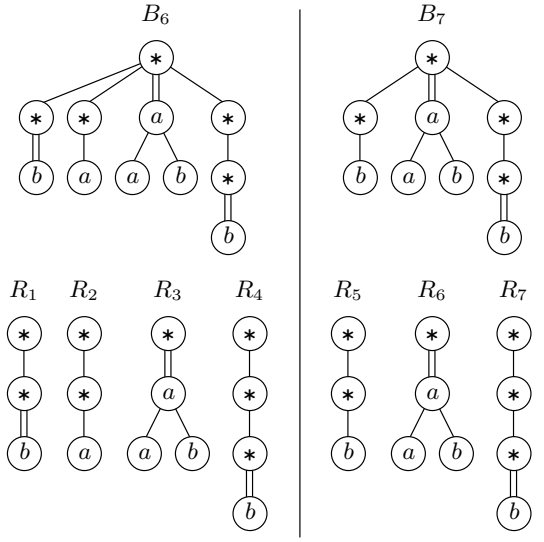

Figure 3: Examples of w-redundancy of branches

The above example shows that we cannot obtain a result about w-redundant branches that is analogous to Proposition 3.8 by replacing $R' \sqsubseteq R$ with $R' \sqsubseteq^{w} R$. To show a similar characterization for w-redundant branches, we need to define a new type of containment.

Definition 3.11 (Relative Containment). *Suppose that B, B_1 and B_2 are b-patterns. B_1 is* contained in B_2 relative to B, *denoted by $B_1 \sqsubseteq_B B_2$, if for all trees t,*

$$B_1(t) \Rightarrow B_2(t) \vee B^{w}(t)\,.$$

That is, if there is an embedding of B_1 in t, then there is either an embedding of B_2 in t or a weak embedding of B in t. $B_1 \equiv_B B_2$ denotes that $B_1 \sqsubseteq_B B_2$ and $B_2 \sqsubseteq_B B_1$.

Clearly, $\sqsubseteq_B$ is weaker than containment, that is, $B_1 \sqsubseteq B_2$ implies $B_1 \sqsubseteq_B B_2$. In general, $\sqsubseteq_B$ and weak containment are incomparable. However, we are usually interested in relative containment of the form $R' \sqsubseteq_B R$, where R is a branch of B. In this case, $\sqsubseteq_B$ is stronger than weak containment, that is, $R' \sqsubseteq_B R$ implies $R' \sqsubseteq^{w} R$ (but the opposite implication does not hold). To see why, observe that if there is a weak embedding of B in a tree t, then there is certainly a weak embedding of the branch R of B in t.

Lemma 3.12. *A branch R of a b-pattern B is w-redundant if and only if $R' \sqsubseteq_B R$ for some branch $R' \neq R$ of B.*

Proof. Consider a branch R that corresponds to a child n of $root(B)$ and let $B^{-} = B - n$. First, we assume that $R' \sqsubseteq_B R$ for some branch $R' \neq R$ of B. To prove $B^{-} \equiv^{w} B$, it is sufficient to show that $B^{-} \sqsubseteq^{w} B$ (because the containment $B \sqsubseteq^{w} B^{-}$ is trivial). So, consider a tree t, such that $B^{-}(t) = \mathbf{true}$. R' is a branch of B^{-}, and therefore, there is an embedding of R' in t. By assumption, $R' \sqsubseteq_B R$. Hence, there is either an embedding of R in t or a weak embedding of B in t. In the former case, the embedding of B^{-} in t can be combined with the embedding of R in t to produce an embedding of B in t, thereby showing that $B(t) = \mathbf{true}$. In the latter case, $B^{w}(t) = \mathbf{true}$. It thus follows that $B^{-} \equiv^{w} B$.

Now suppose that $R' \not\sqsubseteq_B R$ for all branches $R' \neq R$ of B. To show that R is not w-redundant, we will construct a

Algorithm REDUCEPATTERN(B)

1: **for all** nodes n of B **do**
2: Let $B' = B^n_\Delta$
3: **while** B' has branches $R_1 \neq R_2$, s.t. $R_1 \sqsubseteq R_2$ **do**
4: Remove the branch R_2 (except for its root n)
5: **if** n has an incoming descendant edge in B **then**
6: **while** B' has branches $R_1 \neq R_2$, s.t. $R_1 \sqsubseteq_{B'} R_2$ **do**
7: Remove the branch R_2 (except for its root n)

Figure 4: Removing redundancy from a b-pattern

tree t, such that $B^-(t) = \textbf{true}$ and $B^w(t) = \textbf{false}$. For each branch $R' \neq R$, we choose a tree $t_{R'}$, such that $R'(t_{R'}) = \textbf{true}$, $R(t_{R'}) = \textbf{false}$ and $B^w(t_{R'}) = \textbf{false}$. (The tree $t_{R'}$ exists because $R' \not\sqsubseteq_B R$.) The tree t is obtained by joining the roots of all the $t_{R'}$. Clearly, $B^-(t) = \textbf{true}$. To show that $B^w(t) = \textbf{false}$, suppose (by way of contradiction) that e is a weak embedding of B in t. If e maps $root(B)$ to $root(t)$, then it is an embedding of the branch R in one of the $t_{R'}$, contradicting the fact that $R(t_{R'}) = \textbf{false}$; otherwise, e is a weak embedding of B in one of the $t_{R'}$, contradicting the fact that $B^w(t_{R'}) = \textbf{false}$. □

EXAMPLE 3.13. *We continue with Example 3.10. The branch R_2 of B_6 is not redundant; however, it is w-redundant because $R_3 \sqsubseteq_{B_6} R_2$. To prove $R_3 \sqsubseteq_{B_6} R_2$, suppose that t is a tree, such that there an embedding e of R_3 in t and $R_2(t) = \textbf{false}$. We have to show that $B^w_6(t) = \textbf{true}$. Since $R_2(t) = \textbf{false}$, the embedding e maps the descendant edge of R_3 to a path that starts at $root(t)$ and has at least 3 edges. Therefore, a weak embedding of B_6 in t is obtained by mapping the root of B_6 to the grandparent of the node $e(n)$, where n is the upper node of R_3 that is labeled with a. Finally, the branches R_3 and R_4 are not w-redundant because $R_3 \not\sqsubseteq^w R_4$ and $R_4 \not\sqsubseteq^w R_3$. As for B_7, none of its branches is w-redundant. In particular, observe that $R_6 \not\sqsubseteq_{B_7} R_5$, although $R_6 \sqsubseteq^w R_5$.*

4. NORMAL FORMS FOR B-PATTERNS

In this section, we define two classes of b-patterns in terms of normal forms. We believe that these classes capture most of the b-patterns that are used in practice. All the b-patterns B of these classes have two useful properties. First, B is non-redundant if and only if it is minimal. Second, B can be minimized by simply removing redundant and (in some cases) w-redundant branches from every subtree of B.

The algorithm REDUCEPATTERN of Figure 4 eliminates redundancy from a b-pattern B as follows. At each node n of B, it removes redundant branches of B^n_Δ. If a descendant edge enters n, then the algorithm also removes w-redundant branches of B^n_Δ. Note that if the nodes of B are processed bottom-up, then the tests of the loops of Lines 3 and 6 are no longer satisfied after each node is visited just once. It can be shown, however, that the order of visiting nodes is not important. The next lemma shows that REDUCEPATTERN is sound in the sense that it preserves equivalence.

LEMMA 4.1. *Let B' be the result of applying* REDUCEPATTERN *to a b-pattern B. Then $B \equiv B'$.*

PROOF. Eliminating redundant branches of B^n_Δ preserves equivalence to the original B^n_Δ. Hence, the whole b-pattern remains equivalent to B. Removing w-redundant branches preserves weak equivalence to the original B^n_Δ, and since a descendant edge enters n, it also preserves equivalence of the whole b-pattern to the original B (see Proposition 2.9). □

Lines 1–4 of REDUCEPATTERN constitute the algorithm of [4, 5]. These four lines alone cannot always produce a non-redundant b-pattern; an example is the b-pattern $B^r_{n\,\Delta}$ of Figure 2. If Lines 5–7 are also applied, all redundancies are removed from that particular b-pattern. In the next section, we prove that if B is in either one of the two classes defined below, then REDUCEPATTERN actually produces a minimal (and not just non-redundant) b-pattern that is equivalent to B. However, the next example shows that the result of REDUCEPATTERN is not always a non-redundant b-pattern.

EXAMPLE 4.2. *Consider the b-patterns B_8 and B'_8 of Figure 5. Clearly, node n of B_8 is redundant. Nevertheless,* REDUCEPATTERN *does not detect this redundancy because it only looks for redundant (rather than w-redundant) branches when visiting node m. The b-pattern B'_8 differs from B_8 by the additional node with the label d. In this case, the node n' is non-redundant (and as shown later, B'_8 is non-redundant). This example illustrates that it is not always enough to look for redundancy based on local properties of the subtrees B^n_Δ; sometimes, we have to apply global considerations.*

4.1 Stable B-Patterns

As said earlier, two b-patterns may be weakly equivalent even if they are not equivalent. Often, however, the two types of equivalence coincide. A b-pattern B is *stable* if for all b-patterns B', it holds that $B \equiv^w B'$ implies $B \equiv B'$. Testing stability is NP-hard, yet decidable (the proof is omitted). However, the next theorem gives sufficient conditions for stability that can be tested efficiently. We believe that these conditions cover many of the b-patterns that are used in practice. We prove this theorem in the next section.

THEOREM 4.3. *A b-pattern B is stable in each of the following cases.*

1. *$label(root(B)) \neq *$.*
2. *Every child n of $root(B)$ satisfies $\Sigma[B^n_\Delta] \subsetneq \Sigma[B]$.*

The next observation shows that stability is preserved under weak equivalence. The proof follows immediately from the definition of a stable b-pattern and the transitivity of weak equivalence (and equivalence).

OBSERVATION 4.4. *If B is stable and $B \equiv^w B'$, then B' is stable.*

4.2 The Normal Forms $NF_{/*}$ and $NF_{//*}$

DEFINITION 4.5 ($NF_{/*}$ AND $NF_{//*}$). *A b-pattern B is in $NF_{/*}$ (resp., $NF_{//*}$) if every non-root node n of B satisfies at least one of the following.*

1. *A child (resp., descendant) edge enters n.*
2. *B^n_Δ is stable.*

3. B^n_Δ is linear.

The next theorem shows that for b-patterns of $\mathrm{NF}_{/*}$ and $\mathrm{NF}_{//*}$, redundancy elimination is the same as minimization.

THEOREM 4.6. *Consider a b-pattern B that is in either $NF_{/*}$ or $NF_{//*}$. If B' is obtained from B by applying* REDUCEPATTERN*, then $B' \equiv B$ and B' is minimal. In particular, if B is non-redundant, then it is minimal.*

We prove the above theorem by means of several lemmas that are given below. But first we note the following. Consider a b-pattern B of $\mathrm{NF}_{/*}$. If a descendant edge enters node n of B, then B^n_Δ is either linear or stable. In the former case, B^n_Δ has no w-redundant branches. In the latter case, a branch of B^n_Δ is w-redundant if and only if it is redundant. Therefore, the first four lines of REDUCEPATTERN are sufficient for minimizing b-patterns of $\mathrm{NF}_{/*}$.

EXAMPLE 4.7. *Consider the b-pattern B'_8 of Figure 5. The subtree $B'^{\,q}_{8\,\Delta}$ is stable because it satisfies the second condition of Theorem 4.3. Therefore, B'_8 is in $NF_{/*}$. Clearly, B'_8 is not changed by* REDUCEPATTERN*, hence it is minimal.*

LEMMA 4.8. *Suppose that $n_1/B_1 \equiv n_2//B_2$. Then there exists a b-pattern B'_2, such that $n_2//B_2 \equiv n_2/B'_2$ and B'_2 is g-isomorphic to B_2.*

PROOF. We will prove the lemma in stages. At first, we will construct a b-pattern n_1/B'_1 that is equivalent to n_1/B_1. Then we will show how to obtain B'_2 from B_2. Finally, we will prove that $n_2//B_2 \sqsubseteq n_2/B'_2 \sqsubseteq n_2//B_2$.

We start with the construction of the b-pattern B'_1. Consider a path p of n_1/B_1 that begins at the root n_1 and consists of only child edges. If p includes a node with a label other than $*$, then by replacing the descendant edge that emanates from n_2 with a path that is longer than p, we will get a canonical model that is a counterexample to $n_2//B_2 \sqsubseteq n_1/B_1$. So, all the nodes of p are labeled with $*$. In particular, $root(B_1)$ is a wildcard node (and so is $root(B_2)$—see Propositions 2.7 and 2.10). Now suppose that p has a maximal length and it ends at the node m (that has the label $*$). Then either m is a leaf or all the outgoing edges of m are descendant edges. In either case, by replacing the incoming edge of m with a descendant edge, we obtain a pattern that is equivalent to n_1/B_1. If we continue this process of replacing child edges with descendant edges, we will eventually replace all the outgoing edges of $root(B_1)$ with descendant edges. B'_1 is the pattern that is obtained at the point. The important features of B'_1 are that $n_1/B'_1 \equiv n_1/B_1$ and all the outgoing edges of $root(B'_1)$ are descendant edges.

The pattern B'_2 is obtained from B_2 by replacing all the outgoing child edges of $root(B_2)$ with descendant edges. We first prove that $n_2//B_2 \sqsubseteq n_2/B'_2$. Clearly, $B_2 \sqsubseteq B'_2$ and, therefore, $n_2//B_2 \sqsubseteq n_2//B'_2$. In addition, $n_2/B'_2 \equiv n_2//B'_2$ because $root(B'_2)$ is a wildcard node and all its outgoing edges are descendant edges. Therefore, $n_2//B_2 \sqsubseteq n_2/B'_2$.

Finally, we prove that $n_2/B'_2 \sqsubseteq n_2//B_2$. We do it by showing that $B'_2 \sqsubseteq B'_1$. As a result, $n_2/B'_2 \sqsubseteq n_1/B'_1 \equiv n_1/B_1 \equiv n_2//B_2$ (note that n_1 and n_2 must have the same label, as implied by Proposition 2.7). So, let t be a canonical model of B'_2. We will show that $B'_1(t)$ is **true**. Let t_s be obtained from t as follows. Each path of t_s that corresponds to an outgoing edge of $root(B'_2)$ is replaced with a single edge. Let v/t_s be obtained from t_s by adding a new root v that matches the label of n_1 (and n_2) and is connected to $root(t_s)$ by an edge. Then $B_2(t_s) =$ **true** and, therefore, $n_2//B_2(v/t_s) =$ **true**. We conclude that $n_1/B_1(v/t_s) =$ **true** and, therefore, $n_1/B'_1(v/t_s) =$ **true** and so $B'_1(t_s) =$ **true**. In particular, there is an embedding of each branch of B'_1 in some branch of t_s. Since all of the outgoing edges of $root(B'_1)$ are descendant edges, $B'_1(t) =$ **true** (and, actually, an embedding of B'_1 in t_s is also an embedding of B'_1 in t). It follows that $B'_2 \sqsubseteq B'_1$, as claimed. □

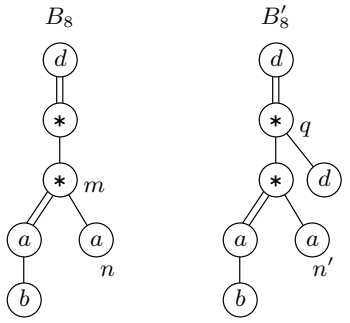

Figure 5: B-patterns B_8 and B'_8 used for illustrating the limitations of REDUCEPATTERN

LEMMA 4.9. *If B is in $NF_{/*}$ (resp., $NF_{//*}$) and B' is the result of executing* REDUCEPATTERN*(B), then B' is in $NF_{/*}$ (resp., $NF_{//*}$).*

PROOF. We prove that every node n of B' satisfies Definition 4.5 in exactly the same way that it satisfies it as a node of B. REDUCEPATTERN does not change the type of edges or add new children to existing nodes. So, it remains to deal with the case of a non-root node n of B', such that B^n_Δ is stable. The algorithm deletes from B^n_Δ only w-redundant nodes. Therefore, $B'^{\,n}_{\Delta} \equiv^w B^n_\Delta$. By Observation 4.4, it follows that $B'^{\,n}_{\Delta}$ is stable. □

The next lemma shows that the output of REDUCEPATTERN is a minimal b-pattern provided that it is in $\mathrm{NF}_{/*}$.

LEMMA 4.10. *Consider a b-pattern B of $NF_{/*}$ that is not changed by* REDUCEPATTERN*. If $B' \equiv B$, then B is g-isomorphic to a subtree of B' that has the same root as B'. Therefore, B is minimal.*

PROOF. The proof is by induction on the height of B. The basis of the induction is trivial because B is a single node. Next, we assume that B has at least two nodes and prove the inductive step.

We first remove redundant branches from B'. (By proving the induction hypothesis after removing those branches, we show that it holds for the original B'.) Let r and r' be the roots of B and B', respectively. Note that r and r' have the same label because $B \equiv B'$. In [4], the following is shown for equivalent b-patterns B and B' without redundant branches. There is a bijection φ from the children of r to those of r', such that for each child n of r, $B^r_{n\,\Delta} \equiv B'^{\,r'}_{\varphi(n)\,\Delta}$. We will show that for all children n of r, the branch $B^r_{n\,\Delta}$ is g-isomorphic to a subtree of $B'^{\,r'}_{\varphi(n)\,\Delta}$ that is rooted at r'. So, consider a child

n of r and let $n' = \varphi(n)$. By the assumptions of the lemma, B^n_Δ is not changed by ReducePattern and is in $\mathrm{NF}_{/*}$.
Case 1: (r, n) is a child edge. By Lemma 4.8, we can assume that (r', n') is also a child edge (otherwise, we replace the branch ${B'}^{r'}_{\Delta\, n'}$ with an equivalent and g-isomorphic branch). Therefore, $B^n_\Delta \equiv {B'}^{n'}_\Delta$. By the inductive hypothesis, B^n_Δ is g-isomorphic to a subtree of ${B'}^{n'}_\Delta$ that is rooted at n'.
Case 2: (r, n) is a descendant edge and B^n_Δ is linear. By Proposition 2.6, $B^r_{\Delta\, n}$ and ${B'}^{r'}_{\Delta\, n'}$ have the same height. Thus, $B^r_{\Delta\, n}$ is g-isomorphic to a path of ${B'}^{r'}_{\Delta\, n'}$ that starts at r'.
Case 3: (r, n) is a descendant edge and B^n_Δ is nonlinear. By the definition of $\mathrm{NF}_{/*}$, B^n_Δ is stable. By Proposition 2.10, $B^n_\Delta \equiv^w {B'}^{n'}_\Delta$. Therefore, $B^n_\Delta \equiv {B'}^{n'}_\Delta$. By the induction hypothesis, B^n_Δ is g-isomorphic to a subtree of ${B'}^{n'}_\Delta$ that is rooted at n'. □

Lemmas 4.1, 4.9 and 4.10 prove Theorem 4.6 for patterns of $\mathrm{NF}_{/*}$. Next, we prove it for patterns of $\mathrm{NF}_{//*}$.

Lemma 4.11. *Consider b-patterns B and B' with roots r and r', respectively. If B and B' have no w-redundant branches, then the following are equivalent.*

1. $B \equiv^w B'$
2. *There exists a bijection φ from the children of r to those of r', such that for all children n of r,*

$$B^r_{\Delta\, n} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, \varphi(n)} \text{ and } {B'}^{r'}_{\Delta\, \varphi(n)} \sqsubseteq_B B^r_{\Delta\, n}.$$

Proof. We first prove that Condition 1 implies Condition 2. We assume that $B \equiv^w B'$ and construct the mapping φ. Let n be a child of r. The mapping $\varphi(n)$ is defined as follows. Let t_n be a tree such that $(B - n)(t_n) = \mathbf{true}$ and yet there is no weak embedding of B in t_n (if no such t_n exists, then n is w-redundant). We choose as $\varphi(n)$ an arbitrary child n' of r' that satisfies ${B'}^{r'}_{\Delta\, n'}(t_n) = \mathbf{false}$. Node n' exists because $B \equiv^w B'$ and $B^w(t_n) = \mathbf{false}$.

Now, we show that for all children n of r, $B^r_{\Delta\, n} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, \varphi(n)}$. So, let n be given and t be a tree, such that $B^r_{\Delta\, n}(t) = \mathbf{true}$. Let t^+_n be obtained from t_n and t by merging their roots. Then $B(t^+_n) = \mathbf{true}$, and as a result, there is a weak embedding e of B' in t^+_n. If e is an embedding (i.e., $e(r) = r'$), then e induces an embedding of ${B'}^{r'}_{\Delta\, n'}$ in t because ${B'}^{r'}_{\Delta\, n'}(t_n) = \mathbf{false}$. Otherwise, the image of e must be contained in t (because there is no weak embedding of B' in t_n), and therefore, e is a weak embedding of B' in t. It follows that there is either an embedding of ${B'}^{r'}_{\Delta\, n'}$ in t or a weak embedding of B' in t. We conclude that $B^r_{\Delta\, n} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, \varphi(n)}$, as claimed.

We now show that φ is a bijection and ${B'}^{r'}_{\Delta\, \varphi(n)} \sqsubseteq_B B^r_{\Delta\, n}$ for all children n of r. Let φ' be a mapping that is constructed, as described above, from the children of r' to those of r. It suffices to show that both φ and φ' are injections and $\varphi' = \varphi^{-1}$. So, suppose otherwise. Then there are two children $n_1 \neq n_2$ of r and a child n' of r', such that $\varphi(n_1) = n'$ and $\varphi'(n') = n_2$ (or there is a symmetric case that is handled similarly). Therefore, $B^r_{\Delta\, n_1} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, n'}$ and ${B'}^{r'}_{\Delta\, n'} \sqsubseteq_B B^r_{\Delta\, n_2}$. The relation $\sqsubseteq_B$ is transitive and $B \equiv^w B'$. Hence, $B^r_{\Delta\, n_1} \sqsubseteq_B B^r_{\Delta\, n_2}$. By Lemma 3.12, this contradicts the fact that B has no w-redundant branches.

Next, we prove that Condition 2 implies Condition 1, namely, $B \sqsubseteq^w B'$ (weak containment in the other direction is proved similarly). Let t be a tree, such that $B(t) = \mathbf{true}$. Hence, $B^r_{\Delta\, n}(t) = \mathbf{true}$ for all children n of r. If ${B'}^{r'}_{\Delta\, \varphi(n)}(t) = \mathbf{true}$ for all n, then $B'(t) = \mathbf{true}$; otherwise, by Condition 2, there is a weak embedding of B' in t. □

We are now ready to prove Theorem 4.3 that gives sufficient conditions for stability.

Proof of Theorem 4.3. Observe that each of the two conditions must hold for every canonical model of B, assuming that it is true for B. A label is *simple* if it is neither $*$ nor $\perp$. The *rank* of a tree is the maximum number of simple labels that appear on any path from the root to a leaf. Observe that a b-pattern and each of its canonical models have the same rank. Furthermore, a weak embedding can map a b-pattern only to a tree that has at least the same rank. Consequently, two weakly equivalent b-patterns have the same rank.

Now suppose that B satisfies the first condition of the theorem, and $B \equiv^w B'$. By Proposition 2.7, B' also satisfies this property. Since weakly equivalent b-patterns and their canonical models have the same rank, it follows that a weak embedding e of B' in a canonical model t of B must map $root(B')$ to $root(t)$; that is, e is actually an embedding. The same is true for a weak embedding of B in a canonical model of B'. Thus, we conclude that $B \equiv B'$, thereby showing that B is stable.

Now consider a b-pattern B that satisfies the second condition, and suppose that $B \equiv^w B'$. We first show that $B \sqsubseteq B'$. So, let t be a canonical model of B. Since the second condition of the theorem holds for B, it must also hold for t. There is a weak embedding e of B' in t because $B \equiv^w B'$. Suppose that e does not map $root(B')$ to $root(t)$. Then there is a symbol of Σ that appears in B but not in B', in contradiction to Proposition 2.8. Thus $B \sqsubseteq B'$, as claimed.

To complete the proof, we show that B' must also satisfy the second condition of the theorem. Then, by symmetry, we conclude that $B' \sqsubseteq B$. Let $r = root(B)$, $r' = root(B')$ and n' be a child of r'. Let B'' be obtained from B' by removing all the w-redundant branches. By Lemma 3.12, $root(B'')$ has a child n'', such that ${B'}^{r'}_{\Delta\, n''} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, n'}$ (note that if n' is in B'', then $n' = n''$). By Lemma 4.11, r has a child n, such that $B^r_{\Delta\, n} \equiv_{B'} {B'}^{r'}_{\Delta\, n''}$. Hence, $B^r_{\Delta\, n} \sqsubseteq_{B'} {B'}^{r'}_{\Delta\, n'}$ and, as a result, $B^r_{\Delta\, n} \sqsubseteq^w {B'}^{r'}_{\Delta\, n'}$. By Propositions 2.10, $B^n_\Delta \sqsubseteq^w {B'}^{n'}_\Delta$, and by Proposition 2.8, $\Sigma[{B'}^{n'}_\Delta] \subseteq \Sigma[B^n_\Delta]$. So, $\Sigma[{B'}^{n'}_\Delta] \subsetneq \Sigma[B] = \Sigma[B']$, as claimed. □

The next lemma shows that the output of ReducePattern is a minimal b-pattern provided that it is in $\mathrm{NF}_{//*}$. The proof of Theorem 4.6 for $\mathrm{NF}_{//*}$ follows from Lemma 4.1, Lemma 4.9 and the next one. Note that the second part in the assumption about B' (in the lemma below) is needed for the inductive hypothesis.

Lemma 4.12. *Consider a b-pattern B of $NF_{//*}$ that is not changed by* ReducePattern. *Suppose that B' is any*

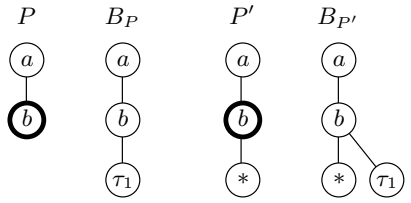

Figure 6: Illustrating a wrong reduction of g-patterns to b-patterns

b-pattern, such that either 1. $B \equiv B'$, or 2. $B \equiv^w B'$ and B has no w-redundant branches (that is, $*//B$ is not changed by ReducePattern*). Then there is a g-isomorphism between B and a subtree of B' that has the same root as B'.*

Proof. The proof is by induction on the height of B. If the height is 0, then the lemma is trivial. For the inductive step, consider a B that has a height $h \geq 1$ and suppose that the lemma is true for height $h-1$. We prove both parts of the lemma simultaneously. We start by removing branches of B' as follows. For proving Part 1, we remove redundant branches, and for Part 2, we remove w-redundant branches. By proving the induction hypothesis after removing those branches, we show that it holds for the original B'.

Let r and r' be the roots of B and B', respectively. We use the bijection φ from the children of r to those of r' that exists for Part 1 as described in the proof of Lemma 4.10 and exists for Part 2 by Lemma 4.11. Consider a child n of r and let $n' = \varphi(n)$. Next, we prove that $B^r_{n\,\Delta}$ is g-isomorphic to a subtree of $B'^{r'}_{n'\,\Delta}$ that is rooted at r'. Note that by the properties of φ (in the proof of either Part 1 or 2), it holds that $B^r_{n\,\Delta} \equiv^w B'^{r'}_{n'\,\Delta}$ and, hence, $B^n_\Delta \equiv^w B'^{n'}_\Delta$. The rest of the proof is the same for both parts of the lemma.

Case 1: (r, n) is a descendant edge. In this case, $*//B^n_\Delta$ is not changed by ReducePattern because neither is B. By the inductive hypothesis of Part 2, B^n_Δ is g-isomorphic to a subtree of $B'^{n'}_\Delta$ that is rooted at n'.

Case 2: (r, n) is a child edge and B^n_Δ is linear. This case is handled as the previous one because a linear b-pattern does not have w-redundant branches.

Case 3: (r, n) is a child edge and B^n_Δ is nonlinear. By the definition of $\mathrm{NF}_{//*}$, B^n_Δ is stable and, hence, $B^n_\Delta \equiv B'^{n'}_\Delta$. By the inductive hypothesis of Part 1, B^n_Δ is g-isomorphic to a subtree of $B'^{n'}_\Delta$ that is rooted at n'. □

In [5], they study the sub-fragment consisting of all b-patterns B, such that wildcard nodes of B have at most one child. They show that their algorithm (i.e., Lines 1–4 of ReducePattern) can minimize all the b-patterns B of that sub-fragment. Although B is not necessarily in $\mathrm{NF}_{/*}$, we can transform it into an equivalent b-pattern of $\mathrm{NF}_{/*}$ by repeatedly applying the following operation. If a descendant edge enters a node n labeled with $*$, then we replace the incoming edge of n with a child edge and also replace the outgoing edge (if it exists) with a descendant edge. Note that this operation is well defined and preserves equivalence because n is labeled with $*$ and has at most one child. Clearly, for all subtrees B^m_Δ of B that have more than one branch, the above operation does not create new redundant branches (because it replaces a branch of B^m_Δ with an equivalent one). Therefore, it suffices to apply Lines 1–4 of ReducePattern to the original b-patterns B (rather than firstly transforming B into $\mathrm{NF}_{/*}$). Thus, the above result of [5] follows from the properties of $\mathrm{NF}_{/*}$.

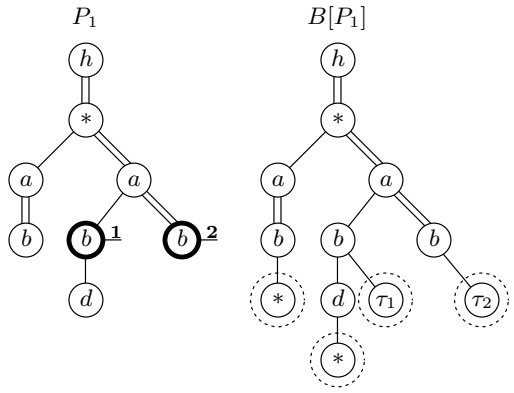

Figure 7: An example of a b-version

5. NORMAL FORMS FOR G-PATTERNS

A *generating pattern* (*g-pattern* for short) has one or more output nodes. We now extend the results of the previous section to g-patterns. At first, we discuss how to reduce g-patterns to b-patterns.

We assume that there is an infinite list of labels $\tau_1, \tau_2, \ldots$ of Σ, such that patterns and trees do not contain any τ_i unless explicitly stated otherwise.

If P is a g-pattern with $out(P) = (o_1, \ldots, o_k)$, then we sometimes write $P^{(k)}$ to explicitly specify the arity of $out(P)$.

5.1 On Reducing G-Patterns to B-Patterns

Given a g-pattern P with $out(P) = (o_1, \ldots, o_k)$, we construct the b-pattern B_P as follows. For $i = 1, 2, \ldots, k$, we add to o_i a new child edge that leads to a node labeled with τ_i. Note that an output node gets as many new children as its number of occurrences in $out(P)$. In [6], it was claimed that $P^{(k)} \sqsubseteq P'^{(k)}$ if and only if $B_{P^{(k)}} \sqsubseteq B_{P'^{(k)}}$ (thus, reducing containment of g-patterns to that of b-patterns). The next example shows that this claim is wrong.

Example 5.1. *Consider the patterns P and P' of Figure 6. Clearly, $P \not\sqsubseteq P'$ because P' requires the output node not to be a leaf. Nevertheless, $B_P \sqsubseteq B_{P'}$.*

Next, we modify the reduction of [6] and use it to derive results about redundancy and minimization of g-patterns.

5.2 B-Versions of G-Patterns

Let P be a g-pattern with $out(P) = (o_1, \ldots, o_k)$. The *b-version* of P, denoted $B[P]$, is obtained from P as follows. First, for all $1 \leq i \leq k$, we add a child edge (o_i, m_i), where m_i is a new node with the label τ_i. Second, for each leaf $n \notin \{o_1, \ldots, o_k\}$, we add a child edge (n, m), where m is a new wildcard node. As an example, Figure 7 shows the g-pattern P_1 and its b-version $B[P_1]$. The newly added nodes are surrounded by dotted circles.

The next proposition shows that a correct reduction from containment of g-patterns to that of b-patterns is by means of replacing g-patterns with their b-versions.

PROPOSITION 5.2. $P_1^{(k)} \sqsubseteq P_2^{(k)} \Leftrightarrow B[P_1^{(k)}] \sqsubseteq B[P_2^{(k)}]$.

PROOF. We first prove the left-to-right implication. Suppose that $P_1 \sqsubseteq P_2$. We consider a tree t, such that $B[P_1](t) =$ **true**, and show that $B[P_2](t) =$ **true**. Let t^- be the tree that is obtained from t by pruning all the leaves. Suppose that e is an embedding of $B[P_1]$ in t, and let e^- be the restriction of e to the nodes of P_1. Note that none of the nodes of P_1 is mapped to a leaf of t. Therefore, e^- is an embedding of P_1 in t^-. For $1 \leq i \leq k$, let $v_i = e(o_i)$, where o_i is the ith output node of P_1. Since $P_1 \sqsubseteq P_2$, there is an embedding f^- of P_2 in t^- that maps the output nodes of P_2 to $v_1, \ldots, v_k$, respectively. We obtain an embedding f of $B[P_2]$ in t by extending f^- as follows. Let n be a leaf of $B[P_2]$ and m be its parent. If $label(n) = *$, then we map n to an arbitrary child of $f^-(m)$. Otherwise, there is a τ_i, such that $label(n) = \tau_i$; that is, m is the ith output node of P_2 and, hence, $f^-(m) = e(o_i) = v_i$. So, we map n to the child of v_i that is labeled with τ_i (this child exists because $B[P_1]$ has a child edge (o_i, m_i), where m_i is labeled with τ_i, and e is an embedding of $B[P_1]$ in t).

We now prove the other direction. Suppose that $B[P_1] \sqsubseteq B[P_2]$ and let e be an embedding of P_1 in a tree t. Note that we assume that t does not contain the labels $\tau_1, \ldots, \tau_k$. Let o_i^1 and o_i^2 be the ith ($1 \leq i \leq k$) output nodes of P_1 and P_2, respectively. We now construct an embedding f of P_2 in t, such that $f(o_i^2) = e(o_i^1)$. Let t^+ be obtained from t as follows. For each leaf of t, we add a child labeled with l, where $l \notin \{\tau_1, \ldots, \tau_k\}$. In addition, for all $1 \leq i \leq k$, we add a child labeled with τ_i to $e(o_i^1)$. So, e can be extended to an embedding of $B[P_1]$ in t^+, that is, $B[P_1](t^+) =$ **true**. Therefore, $B[P_2](t^+) =$ **true** because we assumed that $B[P_1] \sqsubseteq B[P_2]$. Let f^+ be an embedding of $B[P_2]$ in t^+. Since f^+ maps none of the nodes of P_2 to a leaf of t^+, it follows that f^+ includes an embedding f of P_2 in t. Furthermore, for all $1 \leq i \leq k$, the node $e(o_i^1)$ is the only node of t^+ that has a child labeled with τ_i, hence $f(o_i^2) = e(o_i^1)$. □

The following theorem shows that the reduction from a g-pattern to its b-version preserves non-redundancy.

THEOREM 5.3. *P is non-redundant if and only if $B[P]$ is non-redundant.*

PROOF. Let $out(P) = (o_1, \ldots, o_k)$. First, suppose that $B[P]$ is redundant. Hence, there is a proper subtree B' of $B[P]$, such that $B' \equiv B[P]$. Let P' be obtained from B' by pruning all the leaves. If all the nodes of P are in B', then some leaf n of P is also a leaf of B' and, so, n is removed when constructing P'. Therefore, P' is a proper subtree of P. Next, we prove that $P' \sqsubseteq P$ and, consequently, $P' \equiv P$.

Note that B' contains all the nodes labeled with $\tau_1, \ldots, \tau_k$ because each of these labels has a unique occurrence in $B[P]$. Therefore, P' contains all the output nodes of P. Now suppose that e is an embedding of P' in a tree t that does not contain the labels $\tau_1, \ldots, \tau_k$. We construct an embedding f of P in t, such that $e(o_i) = f(o_i)$ for all $1 \leq i \leq k$. Let t^+ be obtained from t as follows. For each leaf n of B' with a parent m (note that $m \in \mathcal{N}(P')$), we add to $e(m)$ a new child with the label $label(n)$ (or some arbitrary label $l \notin \{\tau_1, \ldots, \tau_k\}$ if $label(n) = *$). Then there is an embedding of B' in t^+ and, therefore, there is an embedding f^+ of $B[P]$ in t^+. Note that f^+ cannot map a node of P to a leaf of t^+. Therefore, we can restrict f^+ to an embedding f of P in t. Furthermore, each o_i is mapped by f to $e(o_i)$ because $e(o_i)$ is the only node that has a child with the label τ_i. This concludes the proof of the first direction.

Now suppose that P has a proper subtree P', such that $P' \equiv P$. By definition, the output nodes of P' are the same as those of P. We need to show that $B[P]$ has a proper subtree B', such that $B' \sqsubseteq B[P]$. By Proposition 5.2, $P' \sqsubseteq P$ implies that $B[P'] \sqsubseteq B[P]$. Note that $B[P']$ is not necessarily a subtree of $B[P]$, but the former has fewer nodes than the latter because P' is a proper subtree of P. So, we construct a subtree B' of $B[P]$ that has the same size as $B[P']$ and satisfies $B' \sqsubseteq B[P']$ (hence, $B' \equiv B[P]$).

To obtain B', we start with P'. For each $1 \leq i \leq k$, we add the edge (o_i, n_i), where n_i is the unique node of $B[P]$ with the label τ_i. For each leaf n of P' that is not an output node, either n has a child in P or n is a leaf in P and has a child in $B[P]$. So, n has some outgoing edges in $B[P]$ and we add one of those to B'. Observe that $B' \sqsubseteq B[P']$ for the following reason. $B[P']$ can be obtained from B' by changing, in some leaves, the label and incoming edge into $*$ and a child edge, respectively.[2] □

Algorithm REDUCEGPATTERN(P)

1: $B \leftarrow B[P]$
2: REDUCEPATTERN(B)
3: Prune all the leaves of B
4: $P \leftarrow B$ (with the original output nodes)

Figure 8: Removing redundancy from a g-pattern

In general, it is unknown whether a g-pattern is minimal if and only if its b-version is minimal. However, the next section shows that this is true for large classes of g-patterns.

5.3 Normal Forms of G-Patterns

We now define the versions of $\mathrm{NF}_{/*}$ and $\mathrm{NF}_{//*}$ for g-patterns.

DEFINITION 5.4. *A g-pattern P is in $NF_{/*}$ (resp., $NF_{//*}$) if $B[P]$ is in $NF_{/*}$ (resp., $NF_{//*}$).*

Next, we will prove that for a g-pattern in $\mathrm{NF}_{/*}$ or $\mathrm{NF}_{//*}$, the algorithm REDUCEGPATTERN of Figure 8 eliminates all the redundant nodes and, moreover, minimizes the given g-pattern. This algorithm accepts a g-pattern P as input, constructs $B = B[P]$ (which is in $\mathrm{NF}_{/*}$ or $\mathrm{NF}_{//*}$), minimizes it using REDUCEPATTERN and removes the leaves from the result. For the proof, we need the following observation.

OBSERVATION 5.5. *If B' is obtained from B by executing* REDUCEPATTERN(B), *then $leaves(B') \subseteq leaves(B)$.*

PROOF. REDUCEPATTERN changes the input b-pattern B only in the following way (Lines 4 and 7). It removes from B a whole subtree B^m_Δ, where the parent of m has at least two children. Therefore, if n is not a leaf of B, then it cannot become a leaf during the execution of the algorithm. □

LEMMA 5.6. *Let P be a g-pattern in $NF_{/*}$ (resp., $NF_{//*}$). If P' is the result of executing* REDUCEGPATTERN(P), *then P' is in $NF_{/*}$ (resp., $NF_{//*}$).*

[2]Converting a descendant edge of a b-pattern to a child edge preserves equivalence if that edge enters a wildcard leaf.

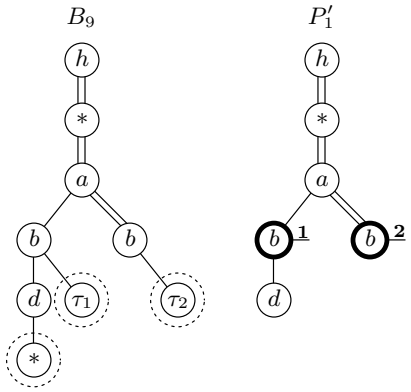

Figure 9: Restoring a g-pattern from a minimized b-pattern

PROOF. Assume that P is in $\mathrm{NF}_{/*}$. The proof for $\mathrm{NF}_{//*}$ is similar. Let B' be the result of applying Line 2 of REDUCEGPATTERN(P). By Observation 5.5, $B[P']$ is isomorphic to B' (because the leaves that are removed from B' in Line 3 are added to P' when constructing $B[P']$). Line 2 is an execution of REDUCEPATTERN and, by Lemma 4.9, it preserves membership in $\mathrm{NF}_{/*}$. We conclude that after executing Line 2, B' is in $\mathrm{NF}_{/*}$ and, therefore, so is P'. □

LEMMA 5.7. *Suppose that $P \equiv P'$, where P is a non-redundant g-pattern in $NF_{/*}$ or $NF_{//*}$. P is g-isomorphic to a subtree of P'.*

PROOF. Assume that P is in $\mathrm{NF}_{/*}$ (the proof for $\mathrm{NF}_{//*}$ is similar). Hence, $B[P]$ is in $\mathrm{NF}_{/*}$. By Theorem 5.3, $B[P]$ is non-redundant. Furthermore, by Proposition 5.2, $B[P] \equiv B[P']$. By Lemma 4.10, $B[P]$ is g-isomorphic to a subtree of $B[P']$. It thus follows that P is g-isomorphic to a subtree of P', as claimed. □

THEOREM 5.8. *The following hold for a g-pattern P in $NF_{/*}$ or $NF_{//*}$.*

1. *P is non-redundant if and only if P is minimal.*
2. REDUCEGPATTERN(P) *results in a minimal g-pattern that is equivalent to P.*

PROOF. Part 1 follows immediately from Lemma 5.7. So, we prove Part 2. Let P' be the result of applying REDUCEGPATTERN(P), and let B' be the result of executing Line 2. By Observation 5.5, $B[P']$ is isomorphic to B'. By Theorem 4.6, $B[P']$ is non-redundant and, by Theorem 5.3, so is P'. Lemma 5.6 shows that P' is in $\mathrm{NF}_{/*}$ or $\mathrm{NF}_{//*}$. Hence, Part 1 implies that P' is minimal. Lemma 4.1 and Proposition 5.2 imply that $P \equiv P'$. □

EXAMPLE 5.9. *Consider again the patterns P_1 and $B[P_1]$ of Figure 7. Note that P_1 is in $NF_{//*}$ because so is $B[P_1]$. By applying* REDUCEPATTERN *to eliminate redundancy from $B[P_1]$, we obtain the b-pattern B_9 of Figure 9. Finally, by pruning the leaves of B_9, we get the g-pattern P'_1 (also in Figure 9). Since P_1 is in $NF_{//*}$, we conclude that P'_1 is a minimal g-pattern that is equivalent to P_1.*

6. COMPLEXITY RESULTS

In this section, we prove that determining whether a pattern is redundant and whether it is w-redundant are coNP-complete problems. We also show that testing whether a pattern is minimal (or w-minimal) is NP-hard.

PROPOSITION 6.1. *The problem of testing whether a pattern is non-redundant (or not w-redundant) is in NP.*

PROOF. We first prove the claim for non-redundancy of a b-pattern B. By Proposition 3.3, to determine that B is non-redundant, we have to show that for all leaves n of B, $B{-}n \not\sqsubseteq B$. In [6], the following is proved. First, testing whether $B(t) = \mathbf{true}$ is in polynomial time in B and t. Second, if $B_1 \not\sqsubseteq B_2$, then there is a tree $t_1 \in \mathit{CMod}(B_1)$, such that $B_2(t_1) = \mathbf{false}$ and $|\mathcal{N}(t_1)| = \mathcal{O}(\mathrm{size}(B_1) \cdot \mathrm{size}(B_2))$. Hence, an efficiently verifiable evidence that B is non-redundant comprises a polynomial-size tree $t_n \in \mathit{CMod}(B{-}n) \setminus \mathit{Mod}(B)$ for each leaf n of B.

By the proof for the case of a b-pattern and Theorem 5.3, testing whether a pattern P is non-redundant is in NP.

Testing non-w-redundancy is in NP because a pattern P is not w-redundant if and only if $*//P$ is non-redundant. □

The intractability results below are for b-patterns. But their proof is immediately applicable to g-patterns that have the root as the only output node (in particular, weak containment and containment are the same in this case). Hence, we get the same intractability results for g-patterns.

Next, we prove coNP-hardness. In [6], Miklau and Suciu show that determining containment of b-patterns is coNP-complete. Our proof of hardness is based on their reduction from the 3SAT problem. For a 3-CNF formula ψ, they construct two b-patterns B_ψ and B'_ψ, such that $B_\psi \sqsubseteq B'_\psi$ if and only if ψ is not satisfiable. The roots of B_ψ and B'_ψ have the same label and, so, we join these roots to obtain a new b-pattern B^{MS}_ψ (see Figure 10 for an illustration). The new b-pattern has the following properties for all 3-CNF formulae ψ. First, B^{MS}_ψ is in both $\mathrm{NF}_{/*}$ and $\mathrm{NF}_{//*}$ because every wildcard node has exactly one child, and that child is a leaf. Second, every subtree $(B^{\mathrm{MS}}_\psi)^n_\Delta$ has no redundant branches, except possibly in the following cases.

- $\mathit{label}(n) \neq *$ and all the branches of n are linear and each has 4 or fewer nodes.[3]
- n is the root of B^{MS}_ψ.

In the first case, we can use Proposition 3.8 to efficiently remove redundant branches, and the resulting b-pattern remains in both normal forms. In the second case, n has two branches which are B_ψ and B'_ψ. The branch B'_ψ is not contained in B_ψ, whereas $B_\psi \sqsubseteq B'_\psi$ if and only if ψ is not satisfiable. Thus, we get the following lemma.

LEMMA 6.2. *For all 3-CNF formulae ψ, we can efficiently construct a b-pattern $\hat{B}_\psi$ that has the following properties. 1. $\hat{B}_\psi$ is in both $NF_{/*}$ and $NF_{//*}$. 2. $\mathit{label}(\mathit{root}(\hat{B}_\psi)) \neq *$. 3. For all non-root nodes n of $\hat{B}_\psi$, the b-pattern $\hat{B}_\psi{}^n_\Delta$ has no redundant branches. 4. $\hat{B}_\psi$ has a redundant branch if and only if ψ is not satisfiable.*

[3]Specifically, n is a root of (an occurrence of) the pattern V in the reduction of [6].

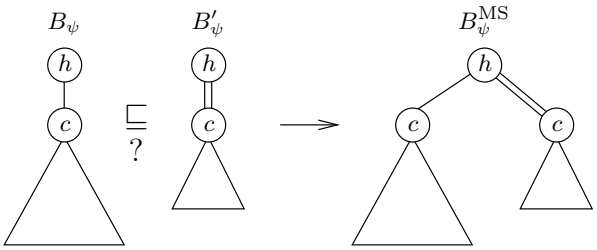

Figure 10: Construction of $B^{\mathbf{MS}}_\psi$

From Proposition 6.1 and Lemma 6.2, we conclude the following theorem.

THEOREM 6.3. *It is NP-complete to decide for a given b-pattern B whether it is non-redundant (or not w-redundant). Testing whether B is minimal (or w-minimal) is NP-hard.*

PROOF. Consider a 3-CNF formula ψ. Recall that only Lines 1–4 of REDUCEPATTERN are needed for minimizing b-patterns of $\mathrm{NF}_{/*}$. Therefore, Properties 1, 3 and 4 of $\hat{B}_\psi$ in conjunction with Lemma 4.10 show that $\hat{B}_\psi$ is minimal if and only if ψ is satisfiable.

$\hat{B}_\psi$ is a stable pattern because of Property 2 and Theorem 4.3. Theorem 4.6 (in conjunction with Property 1) and the stability of $\hat{B}_\psi$ imply that the following are equivalent. 1. $\hat{B}_\psi$ is non-redundant. 2. $\hat{B}_\psi$ is not w-redundant. 3. $\hat{B}_\psi$ is minimal. 4. $\hat{B}_\psi$ is w-minimal. □

Note that the above theorem holds even if B satisfies stricter definitions of both $\mathrm{NF}_{/*}$ and $\mathrm{NF}_{//*}$, namely, the stability requirement is replaced with the first sufficient condition of Theorem 4.3. Since B_ψ and B'_ψ satisfy these stricter definitions, testing containment is coNP-complete even for b-patterns that are in both of these stricter normal forms.

Consider the problem of testing whether a b-pattern is minimal (or w-minimal). Clearly, it is in Π^{p}_2. However, we do not know whether this problem is in NP. Of course, if non-redundancy implies minimality, then this problem is NP-complete.

In [4], they prove that the following problem is coNP-hard.[4] Given a b-pattern B and an integer k, determine whether $\min_{\equiv}(B) \leq k$. However, hardness of this problem does not imply hardness of testing whether a pattern is minimal (or whether it is non-redundant).

7. CONCLUSION

We considered the notions of redundancy and minimization in a well known fragment of XPath. It was shown that contrary to results of earlier work [4], the problem of whether non-redundancy implies minimality is still open in the general case. We proved that for patterns of two normal forms, minimality and non-redundancy coincide, and moreover, minimization is realized by removing redundant and (sometimes) w-redundant branches. The two normal forms can be combined in order to cover additional b-patterns.

Our results are based on the new notions of weak containment, relative containment, w-redundancy and stability. By utilizing these notions, we can eliminate redundancies that cannot be detected by either the algorithm of [4, 5] or the common technique [1, 3, 9, 11] of finding a non-injective homomorphism from the pattern to itself. In fact, it can be shown that the latter is only capable of detecting the type of redundancy considered in the former

Stability of sub-patterns is important for showing that a pattern is in one of the normal forms. We proved that some natural properties of patterns guarantee stability. Therefore, we believe that the normal forms cover many of the patterns that are used in practice. Properties of patterns in the normal forms enabled us to show that testing non-redundancy is NP-complete and testing minimality is NP-hard.

We proved our results for b-patterns and extended them to g-patterns by employing the notion of b-versions. It was shown that b-versions are a useful tool for reducing problems of g-patterns to those of b-patterns. In particular, b-versions correct a minor flaw in the reduction of [6].

Several questions remain open. The main one is whether a non-redundant pattern is always minimal. Another open problem is the relationship between minimality of g-patterns and that of their b-versions. By Theorem 5.3, the answer is trivial if minimality is equivalent to non-redundancy. The third open problem is the exact complexity of stability (we can show that it is decidable and NP-hard).

Acknowledgment

The authors thank the anonymous referees for helpful comments and suggestions.

8. REFERENCES

[1] S. Amer-Yahia, S. Cho, L. V. S. Lakshmanan, and D. Srivastava. Tree pattern query minimization. *VLDB J.*, 11(4), 2002.
[2] E. P. F. Chan. Containment and minimization of positive conjunctive queries in OODB's. In *PODS*, 1992.
[3] A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *STOC*, 1977.
[4] S. Flesca, F. Furfaro, and E. Masciari. On the minimization of Xpath queries. In *VLDB*, 2003.
[5] S. Flesca, F. Furfaro, and E. Masciari. On the minimization of Xpath queries. A paper in press, 2007.
[6] G. Miklau and D. Suciu. Containment and equivalence for a fragment of XPath. *J. ACM*, 51(1), 2004.
[7] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, 1999.
[8] F. Neven and T. Schwentick. On the complexity of XPath containment in the presence of disjunction, DTDs, and variables. *Logical Methods in Computer Science*, 2(3), 2006.
[9] P. Ramanan. Efficient algorithms for minimizing tree pattern queries. In *SIGMOD*, 2002.
[10] P. T. Wood. On the equivalence of xml patterns. In *Computational Logic*, 2000.
[11] P. T. Wood. Minimising simple XPath expressions. In *WebDB*, 2001.
[12] P. T. Wood. Containment for XPath fragments under DTD constraints. In *ICDT*, 2003.
[13] W. Xu and Z. M. Özsoyoglu. Rewriting XPath queries using materialized views. In *VLDB*, 2005.

[4]They also claim that this problem is in coNP; but to prove it, they assume that a non-redundant pattern is minimal.