On the complexity of tree pattern matching

Anca Muscholl, Zhilin Wu
LaBRI, Université Bordeaux 1 *

Tree patterns are a basic type of combinatorial patterns. They have been studied in various research fields of computer science, such as programming languages, theorem proving, symbolic computation, and XML [HO82, KM95, MS02]. In XML community, tree patterns are usually seen as a fragment of XPath language, with the implicit assumption that tree pattern matchings are non-injective mappings.

In this talk, we survey as well as get some new results on the complexity of tree pattern matching problem, i.e. given a tree $T$ and a tree pattern $P$, decide whether there is a matching from $P$ to $T$. We unify the research work that has been done into one framework and consider the following variants of the problem,

- Whether $T$ and $P$ are ordered or unordered;
- Whether the matching of $P$ to $T$ is injective or non-injective;
- Whether or not $T$ contains unbounded data and $P$ contains data comparisons.

Fix a finite alphabet $\Sigma$ in the following.

A ordered tree pattern $P$ is a hepta-tuple $(V, E_d, E_s, r, L_v, L_d, L_s)$, where $(V, E_d, r)$ is a $r$-rooted ordered tree, $V$ is the domain, $E_d$ is the parent-child relation (vertical edges), $E_s$ is the next-sibling relation (horizontal edges), $L_v : V \to \Sigma$ is a node labeling function attaching each node a label from $\Sigma$, $L_d : E_d \to \{|, ||\}$ denoting whether each edge in $E_d$ is a child (“|”) edge or a descendant edge (“||”), and $L_s : E_s \to \{-, \equiv\}$ denoting whether each edge in $E_s$ is a next-sibling edge (“—”) or a to-the-right edge (“≡”) ($y$ is to the right of $x$ if $\exists z_1 z_2$ such that $z_2$ is the right sibling of $z_1$, $x = z_1$ or $x$ is a descendant of $z_1$, and $y = z_2$ or $y$ is a descendant of $z_2$).

Given a $\Sigma$-labeled ordered tree $T$ and an ordered tree pattern $P$, a matching of $P$ to $T$ is a mapping $f$ from the domain of $P$ to the domain of $T$ preserving the root, the node-labels, the labels of the vertical edges (“|” or “||”), and the labels of the horizontal edges (“—” or “≡”). Because of the order between siblings, matchings of ordered tree patterns to ordered trees are in fact injective mappings.

Unordered trees and unordered tree patterns, and their matchings can be defined similarly, but without the linear order between siblings. Matchings of unordered tree patterns to unordered trees are not necessarily injective mappings. An injective matching of an unordered tree pattern $P$ to an unordered tree $T$ is an injective mapping satisfying the conditions for matchings.

For injective unordered tree pattern matchings, some natural additional restrictions can be made.

- **Non-descendant relation preservation:** If two nodes in $P$ are non-descendant to each other, then the images of the two nodes under the injective mapping are also non-descendant to each other in $T$.
- **Lowest-common-ancestor relation preservation:** If node $v$ is the lowest common ancestor of two nodes $v_1, v_2$ in $P$, then the image of $v$ under the injective mapping is the lowest common ancestor of the two images of $v_1, v_2$ (under the injective mapping) in $T$.

For (injective) ordered tree pattern matchings, since they preserve the non-descendant relation in a trivial way because of the preservation of the order between siblings, only the lowest-common-ancestor relation preservation will be taken as an additional restriction.

**Non-injective tree pattern matching (without data)**

Since ordered tree pattern matchings are injective, we only need to consider non-injective unordered tree pattern matchings.

With the non-injective semantics, tree patterns can be seen as a fragment of XPath language. Because XPath model checking is in polynomial time [BK08], non-injective unordered tree pattern matching problem is also in polynomial time.

LaBRI, Université Bordeaux 1, 351 cours de la Libération, F-33405, Talence cedex, France, zhilin02@gmail.com
The polynomial-time complexity of the non-injective unordered tree pattern matching problem can also be proved by a dynamic programming approach.

**Injective tree pattern matching (without data)**

Also by a dynamic programming approach, the (injective) ordered tree pattern matching problem can be shown in polynomial time, no matter whether the matchings preserve the lowest-common-ancestor relation or not. The tree pattern matching problem considered in [HO82] corresponds to the ordered tree pattern matching problem defined above with all edges in \( E_d, E_s \) being respectively child edges (“\(|\)”) and next-sibling (“\(—\)” ) edges, while the more general tree pattern matching problem considered in [Cha02] corresponds to the ordered tree pattern matching problem defined above with all edges in \( E_d, E_s \) being respectively child edges (“\(|\)”) and to-the-right (“\(=\)”) edges. In addition, we show that the ordered tree inclusion problem considered in [KM95], i.e. the problem to decide whether the tree pattern \( P \) can be obtained from the tree \( T \) by deleting vertices from \( T \), corresponds to the (injective) ordered tree pattern matching problem with all the edges in \( E_d, E_s \) of the tree pattern being respectively descendant (“\(|\)”) edges and to-the-right (“\(=\)”) edges; and a restricted version of the ordered tree inclusion problem, i.e. the ordered constrained tree inclusion problem considered in [Val03], the problem to decide whether the pattern \( P \) can be obtained from the tree \( T \) by deleting degree-one and degree-two nodes, corresponds to the (injective) lowest-common-ancestor-preserving ordered tree pattern matching problem, with all edges in \( E_d, E_s \) of the tree pattern being respectively descendant (“\(|\)”) edges and to-the-right (“\(=\)”) edges.

Now consider unordered trees and tree patterns.

By a reduction from the subsequence packing problem [Jia06], we prove that the injective unordered tree pattern matching problem is NP-complete.

We show that the unordered tree inclusion problem considered in [KM95] corresponds to the injective non-descendant-preserving unordered tree pattern matching problem with all edges of the tree pattern being descendant (“\(|\)”) edges. Since unordered tree inclusion problem is NP-complete [KM95], it follows that the injective non-descendant-preserving unordered tree pattern matching problem is also NP-complete.

On the other hand, we show that the constrained unordered tree inclusion problem [Val03] corresponds to the injective lowest-common-ancestor-preserving unordered tree pattern matching problem with all edges in the tree pattern being descendant (“\(|\)”) edges. By a dynamic programming approach, we prove that the injective lowest-common-ancestor-preserving unordered tree pattern matching problem is in polynomial time.

**Data tree pattern matching**

Fix an infinite data domain \( D \).

A (ordered or unordered) data tree is a rooted \( \Sigma \)-labeled (ordered or unordered) tree with data values, i.e. each node \( v \) in the tree has an additional label \( D(v) \in D \).

A (ordered or unordered) data tree pattern is a (ordered or unordered) tree pattern with two additional binary relations \( D_\sim, D_\not\sim \), describing respectively the equality and inequality of the data values of nodes.

It follows from the NP-completeness of the injective unordered data tree pattern matching problem [Dav08] that the data tree pattern matching problem is NP-complete for all its variants, ordered or unordered, injective or non-injective, whether or not preserving the the non-descendant relation or the least-common-ancestor relation.

**References**


