Fast Subsumption Between Rooted Labeled Trees

Olivier Carloni(✉)

Sem Spirit, Montpellier, France
semspirit.contact@gmail.com
http://www.semspirit.com/

Abstract. This paper presents two data structures designed to efficiently query a set of rooted labeled trees (forest) defined in a language based on a relational vocabulary Σ and provided with a set-theoretic semantics and a subsumption relation matching the existential conjunctive fragment of the description logic $\mathcal{ALC}$. Given a tree query with q nodes and a forest with n nodes, after showing the equivalence between subsumption and homomorphism, an $O(q \cdot n)$ algorithm is proposed to compute all homomorphisms/subsumptions from the query to the forest. Then, are presented the two search data structures for faster homomorphism/subsumption retrieval. The first one provides a query time of $O(q)$ for a structure size of $O(2^n)$; and the second one provides a trade-off between the query time of $O(k^2 \cdot q)$ and the structure size of $O(k^2 \cdot |\Sigma| \cdot 2^{\lceil n/k \rceil})$, for a fixed integer k.

1 Introduction

The multiplication of data sources and the raise of data volumes have led to the emergence of many efficient search data structures. The first approaches formerly introduced in [10,11] and still inspiring researchers, are the *trie* (or digital tree) and the binary search tree, each of them providing query time in $O(q)$ and $O(q \cdot \log n)$ for a storage size of $O(n)$ and $O(n \cdot \log n)$ where the query is a sequence of q bits and the data set consists in n bit sequences of fixed size. Those two data structures were the starting point for multidimensional range search data structures. Progressively, it emerged that to provide faster query time, the size of the data structure needs to increase exponentially with the tuples dimension, leading to the so-called *curse of dimensionality*. Thus, proposed search data structures were efficient for a small dimension d: as kD-trees [2] ($O(n^{1-1/d})$ time for $O(n)$ size), range-trees [3] ($O(\log^d n)$ time for $O(n \log^{d-1} n)$ size), and quadtrees [7]. More recently, skip lists and quadtrees were combined to propose an innovative data structure [6] that guarantees with some probability the correctness of the query result set. Since tuples are closed to trees, similar issues have been raised in the fields of hierarchical data management and graph theory. In particular, an efficient method was introduced to query XML documents [4] or graphs [9] with queries defined in the *twig* formalism that captures a small useful fragment of the XML query language. Automata theory has also led to some innovative

G. Li et al. (Eds.): KSEM 2017, LNAI 10412, pp. 552–559, 2017.
DOI: 10.1007/978-3-319-63558-3_47

methods, as the one proposed in [8] that uses pushdown-automata to search a rooted tree query pattern within a rooted data tree in a query time linear in the given tree pattern.

The work presented here is also based on automata theory, with the difference that our automata are *word* finite states automata. The main contribution of the paper is the definition of two data structures, both used to store a forest of n nodes whose trees are defined in a knowledge representation (KR) language relying on a relational vocabulary Σ and equipped with a set-theoretic semantics and a subsumption relation matching the existential conjunctive fragment of the description logic $\mathcal{ALC}$. The two data structures are supplied with an efficient interrogation mechanism that complies with the semantics. Given a query of size q, the first one provides $O(q)$ query time for a size of $O(2^n)$; and the second one makes possible a trade-off depending on k between a query time of $O(k^2 \cdot q)$ and a size of $O(k^2 \cdot |\Sigma| \cdot 2^{\lceil n/k \rceil})$. The paper is organized in three parts: the first section presents the tree KR language, its concrete and abstract syntaxes with the set-theoretic semantics and subsumption relation. Then, is introduced the labeled tree homomorphism (shown equivalent to the subsumption) with a linear time algorithm. The second section defines the notion of tour language of a tree, and establishes the equivalence between tour language containment and homomorphism/subsumption. Finally, the third section makes use of the automata theory machinery to build the aforementioned data structures.

2 Trees, Subsumption and Homomorphism

2.1 Concrete Trees

Given a relational vocabulary Σ, the concrete representation of a labeled rooted tree T (in short *concrete tree* or *tree*) defined on Σ is a tuple $T = (N, r, s)$ where N is the non-empty set of nodes, $r \in N$ is the root and $s : N \times \Sigma \to 2^N$ is the successor function between nodes of N. A node $n \in N$ is a λ-*successor* of a node $p \in N$ iff $n \in s(p, \lambda)$. The successor function s is defined such that the node r is the root and s does not contain any cycle. $|T|$ denotes the size in nodes of the tree T and identity between trees is defined by means of tree isomorphism.

2.2 Abstract Syntax

A string a is an abstract representation of a tree (in short *abstract tree*) defined on the relational vocabulary Σ iff either a is the empty string, $a = \lambda X \varnothing$ or $a = XY$ where $\lambda \in \Sigma$, $\varnothing$ is a special symbol not in Σ and X, Y are abstract trees defined on Σ. Let $T = (N_T, r_T, s_T)$ be a concrete tree, an *abstract representation* α_T of T is either the empty string if N_T is a singleton; or else an abstract tree in the form $\alpha_T = \lambda \alpha_u \varnothing \alpha_V$ such that (1) u is one λ-successor subtree of r_T; and (2) V is the concrete tree resulting from the removal of u among the λ-successor subtrees of r_T in T. Let a be an abstract tree defined on Σ, the *concrete representation* $\gamma(a)$ of a is the concrete tree $\gamma(a) = (N_a, r_a, s_a)$ such that: (1) if $a = \lambda x \varnothing Y$

where x and Y are (possibly empty) abstract trees then $\gamma(a)$ is the copy of $\gamma(Y) = (N_Y, r_Y, s_Y)$ such that $r_a = r_Y$ and $\gamma(x)$ belongs to the λ-successor subtrees of r_a; else (2) if a is the empty string then $N_a = \{r_a\}$ is a singleton and s_a is undefined for the root r_a which is the unique node of $\gamma(a)$.

Semantics. An interpretation structure I of a relational vocabulary Σ is a tuple $I = \langle D, i\rangle$ where D is a non-empty set, called the domain and i is an interpretation function that maps every (relational) symbol from Σ to a subset of $D \times D$; and every abstract tree a defined according to Σ to a subset of D such that (1) if a is empty $i(a) = D$, else (2) if $a = \lambda x \varnothing Y$ then $i(a) = i(Y) \bigcap \{o \in D | (o, o') \in i(\lambda)\, and\, o' \in i(x)\}$ where Y and x are abstract trees defined on Σ. This interpretation is equivalent to the one given to the existential conjunctive fragment of the description logic $\mathcal{ALC}$ [1] (i.e. the fragment limited to existential restriction and intersection operators). The equivalence is proven by considering Σ as role names and by defining a bijective function f that recursively translates an abstract tree $a = \lambda x \varnothing Y$ into its equivalent $\mathcal{ALC}$ concept definition $f(a) = (\exists \lambda f(x)) \sqcap f(Y)$. Let a and b be abstract trees, we say that a *subsumes* b (b is subsumed by a), written $b \sqsupseteq a$ ($a \sqsubseteq b$), iff $i(b) \subseteq i(a)$ for all interpretation structures $I = \langle D, i\rangle$ of Σ.

2.3 Rooted Labeled Tree Homomorphism

A rooted labeled tree homomorphism h from a tree A into a tree B is a function $h : N_A \rightarrow N_B$ from the nodes of A into the nodes of B that maps the roots $h(r_A) = r_B$ of the trees and preserves the successor function such that for all $\lambda \in \Sigma$ and $n, p \in N_A$ if n is a λ-successor of p in A then $h(n)$ is a λ-successor of $h(p)$ in B. $h(A)$ denotes the subtree in B that is the image of A by h.

Algorithm. As shown in [12], given a labeled graph with n nodes and a labeled tree with m nodes, there is an algorithm that computes all homomorphisms from the tree into the graph in $O(mn)$ time. Restricting this problem to two trees A and B, the set $\mathcal{H} = \{h | h = A \rightarrow B\}$ of all homomorphisms from A to B can be computed in time $O(|A| \cdot |B|)$ with the two following steps.

The first step consists in running the following procedure $homs(a, A, X, B)$ that returns the set $\{x \in X$ such that there is an homomorphism $h = A \rightarrow B$ and $h(a) = x\}$:

1. For all λ-successors a_i of the node a in A, for any arbitrary symbol $\lambda \in \Sigma$
2. $\quad C_{x,a_i} \leftarrow \{x_j$ of λ-successors of $x\}$, for every $x \in X$
3. $\quad X_i \leftarrow \bigcup_{x \in E} C_{x,a_i}$
4. $\quad$ let R_i be the result of $homs(a_i, A, X_i, B)$
5. $\quad D_{x,a_i} \leftarrow C_{x,a_i} \cap R_i$, for every $x \in X$
6. $\quad X \leftarrow \{x \in X$ such that $D_{x,a_i} \neq \emptyset\}$
7. return X

The second step for enumerating all the homomorphisms h of $\mathcal{H}$, consists in keeping the D_{x,a_i} sets outside the function *homs*. Doing so, one can build each homomorphism h by browsing recursively the structure D_{x,a_i} in order to select an image $h(a'_i)$ for each node a'_i of A fitting the one selected for its parent a'; following a recursive traversal of A from its root to its leaves. More precisely, $h \in \mathcal{H}$ is exhibited by choosing $h(a) \in homs(a, A, X, B)$, then recursively: if $h(a') = x$ and a'_i is a λ-successor of a' then choose $h(a'_i) \in D_{x,a'_i}$ such that $h(a'_i)$ is a λ-successor of x.

Subsumption and Homomorphism Equivalence. *Let a and b be two abstract trees, a subsumes b iff there exists an homomorphism $h = \gamma(a) \to \gamma(b)$ between their respective concrete representations $\gamma(a)$ and $\gamma(b)$.* This can be shown by recurrence. When a is empty, it is obvious. Then by using the recursive definitions of subsomption/homomorphism we prove that: given a λ symbol of a and its corresponding node n (which is a λ-successor of some other) in $\gamma(a)$, there is a symbol in b equal to λ establishing the subsumption (so far) iff its corresponding node n' in $\gamma(b)$ (which is a λ-successor of some other) is a valid image for n, establishing the homomorphism (so far).

3 Tour Language of a Tree

3.1 Automata Induced by a Tree

INFA *(resp IDFA).* Given an alphabet Σ, an incomplete NFA or INFA *(resp. incomplete DFA or IDFA)* is a tuple $A = (Q, \Sigma, \delta, q_0, F)$ where Q is the set of states, δ the transition function $\delta : Q \times \Sigma \to 2^Q$ (which may not be total) *(resp. $\delta : Q \times \Sigma \to Q$)*, q_0 the initial state and F the final states set. Let δ^* be the function such that given a word $w = \lambda w'$ with $\lambda \in \Sigma$ $\delta^*(x, w) = \delta(x, \lambda)$ when $w' = \epsilon$ or otherwise $\delta^*(x, w) = \bigcup\{\delta^*(x', w') | x' \in \delta(x, \lambda)\}$ *(resp. $\delta^*(x, w) = \delta^*(\delta(x, \lambda), w')$)*. An INFA *(resp. IDFA)* A accepts a word w if $\delta^*(x, w) \cap F \neq \emptyset$ *(resp. $\delta^*(x, w) \in F$)* in a time $O(|w|.|A|^2)$ (resp. $O(|w|)$) and the language recognized by A is the set L_A of all words accepted by A. The notation of δ^* is extended to allow δ^* to take a set $X \subseteq Q$ as parameter such that $\delta^*(X, w) = \bigcup\{\delta^*(x', w) | x' \in X\}$.

Tours, Tour Language and Tour Automaton. A word t is a tour of a tree $T = (N, r, s)$ if t is a finite string and (1) either t is empty, (2) either $t = \lambda t' \varnothing t''$ such that t'' is a tour of T and t' is a tour of a subtree of T starting at one of the λ-successors of the root r of T. The tour language L_T of a tree T is the set of all tours of T. A tour t of a tree T is said to be *eulerian* iff $\gamma(t)$ is isomorphic to T. An abstract representation α_T of a tree T is an eulerian tour. Given a tree $T = (N, r, s)$, let $infa(T) = (Q, \Sigma', \delta, q_0, F)$ be *the INFA induced by* T as the INFA defined on the alphabet $\Sigma' = \Sigma \cup \{\varnothing\}$ such that $Q = N$, $q_0 = r$, $F = \{r\}$ and $\delta(x, \lambda) = X'$ and $\delta(x', \varnothing) = \{x\}$ for all $x' \in X'$ iff $s(x, \lambda) = X'$. *Tour language recognition:* Given a tree $T = (N, r, s)$, the INFA $infa(T)$ recognizes the tour language L_T of T.

3.2 Tour Languages, Containment and Homomorphism

By showing that a tour language L_T of a concrete tree T is equal to the set of every abstract tree t for which an homomorphism exists from $\gamma(t)$ to T, it can be established for two trees T_1 and T_2 that **(1)** *there exists an homomorphism* $h = T_1 \rightarrow T_2$ *iff the language containment* $L_{T_1} \subseteq L_{T_2}$ holds; and **(2)** *given an eulerian tour* e *of* T_1, $e \in L_{T_2}$ iff $L_{T_1} \subseteq L_{T_2}$.

Forest and Tour Language Union. Given a tree T and a forest B, h is an homomorphism $h = T \rightarrow B$ from T into B iff there exists a tree $U \in B$ such that h is an homomorphism $h = T \rightarrow U$ from T to U. The tour language L_B of a forest B is the union of the tour languages L_T of all the trees $T \in B$ in the forest. We extend the definition of $infa$ to forests: given a forest B, the INFA $infa(B)$ recognizing the tour language L_B of the forest B contains an initial state q_0 and the INFAs $infa(T_i)$ with initial states q_0^i of the trees $T_i \in B$ such that there is in $infa(B)$ a λ-transition from q_0 to a state x iff there is in $infa(T_i)$ a λ-transition from q_0^i to this state x. Given a forest B and a tree T, there exists an homomorphism $h = T \rightarrow B$ iff $L_T \subseteq L_B$.

4 Search Data Structures

Given a forest B and a tree T with abstract syntax t, we have: $t \in L_B$ iff $L_T \subseteq L_B$ iff there exists an homomorphism $h = T \rightarrow B$. Thus, the automaton $infa(B)$ accepts t iff there is an homomorphism $h = T \rightarrow B$. However checking for an homomorphism by using $infa(B)$ as well as the algorithm given in Sect. 2.3 is done in a time that strongly depends in the size $|B|$ of the forest. The main benefit of the two following search data structures is to decrease this dependency by reducing the non-determinism when testing membership in $infa(B)$.

4.1 Complete Determinization

The INFA $infa(B)$ is determinized into an IDFA $idfa(B)$ with the so-called *powerset construction* algorithm presented in [13]. Let $infa(B) = (Q_N, \Sigma, \delta_N, q_0, F_N)$ be the INFA for the forest B, the IDFA $idfa(B) = (Q_D, \Sigma, \delta_D, X_0, F_D)$ equivalent to $infa(B)$ is such that **(1)** $X_0 = \{q_0\}$ and $X_0 \in Q_D$, **(2)** if $\lambda \in \Sigma$ and $X \in Q_D$ then $\delta_D(X, \lambda) = \{y | x \in X \, and \, y \in \delta_N(x, \lambda)\}$ and $\delta_D(X, \lambda) \in Q_D$; and finally **(3)** for all $X \in Q_D$, X is an accepting state ($X \in F_D$) in $idfa(B)$ iff at least one of its member x is an accepting state ($x \in F_N$) in $infa(B)$. The size of $idfa(B)$ is $|idfa(B)| = O(2^{|B|})$.

Homomorphism Test. Given a forest B and a tree $A = (N, r, s)$ with abstract syntax (or eulerian tour) a, there is an homomorphism $h = A \rightarrow B$ iff a is accepted by $idfa(B)$. Checking if such an homomorphism h exists is done in time $O(|A|)$.

Homomorphisms Reconstruction. Given an abstract representation $a = \alpha_A$ of a tree A, let f_A be the function that maps an index $0 \leq i \leq |a|-1$ of a symbol $a[i]$ in a with its corresponding node in A. f_A is not injective, thus one node x from A may have several antecedents indices and $f_A^{-1}(x)$ is a set. Let g_a be the bijective function that maps an index $0 \leq i \leq |a|-1$ of a symbol in a with the state $\delta^*(q_0, a[0,i])$ of the run reached by the word $a[0,i]$. Now, μ_A is the function mapping each node x of A with a state $\mu_A(x)$ of the run such that $\mu_A(x) = g_a(i_x)$ where i_x is the largest index in $f_A^{-1}(x)$. Thus, $\mu_A(x)$ is the state of the run that recognizes in a the symbol at the farthest position i_x (from the start of the string) and that corresponds to the node $x = f_A(i_x)$. Each state s of the $idfa(B)$ reached by a string w represents a set of nodes $n(s)$ from B in bijection with the set S of states reached in $infa(B)$ by w. Let ω_s be the function partitionning the nodes of $n(s)$ according to their parents such that for a state s in $idfa(B)$ and a node x in B, $\omega_s(x) = n(s) \cap S_x$ where S_x is the set of successors of x. Let π be the function returning the parent node $p = \pi(x)$ in B of any given successor node x of p in B. The set of homomorphisms $\mathcal{H}$ from A into B can be reconstructed with the following procedure.

Initialisation Step. Let $x = f_A(|a|-1)$ be the node in A corresponding to the last symbol in a leading to the accept state in $\mathcal{A}_B$. Every node x' of the domain $dom(\omega_s)$ of ω_s where $s = \mu_A(x)$ is an image of x according to an homomorphism from A to B. Thus we add in $\mathcal{H}$ as many partial homomorphisms h as there exist nodes $x' \in dom(\omega_s)$ such that $h(x) = x'$. These partial homomorphisms will be completed by the following loop until they become full homomorphisms from A to B. Let $\mathcal{H}'$ be an empty set. At each step, the following loop generates a new set $\mathcal{H}'$ containing further completed copies of the homomorphisms in $\mathcal{H}$; and at the end of the step: $\mathcal{H}'$ replaces $\mathcal{H}$.

Reconstruction Loop. Loop until $\mathcal{H}$ and $\mathcal{H}'$ become identical such that, for every homomorphism $h \in \mathcal{H}$ and every node $x \in A$ where $h(x) = x'$: (1) if x is a successor of p in A then we add in $\mathcal{H}'$ a copy h' of h such that $h'(p)$ is set to the unique parent node $\pi(x')$ of x' in B; (2) if y is a λ-successor of x in A then for all node $y' \in \omega_s(x')$ where $s = \mu_A(y)$ we add in $\mathcal{H}$ a copy h' of h such that $h'(y) = y'$.

Reconstruction Time. If π and ω are precomputed maps using a dichotomic method (as BST) to search the unique parent $\pi(x)$ or the children set $\omega(x)$ of a given node x in B, then the search in the maps is performed in $O(\log |B|)$. Since, the $idfa(B)$ accepts $a = \alpha_A$ in time $O(|A|)$ and at most $O(|A|)$ lookups are done in π and ω maps in order to build each homomorphism of $\mathcal{H}$; then supposing that there are K homomorphisms to be returned, the reconstruction time is $O(K.|A|.\log |B|)$.

4.2 Trade Off Between Space and Time

As shown in [5], given an integer k, an INFA of size n and an input word of size w, there exists a data-structure that provides a trade-off between its size of

$O(k^2 \cdot |\Sigma| \cdot 2^{\lceil n/k \rceil})$ and the time $O(w \cdot k^2)$ needed to simulate the INFA on the input word. This data-structure is an array storing k equal sized arbitrary partitions of the INFA such that inside each partition the simulation is deterministic. The remaining non-determinism occurs when multiple partitions have to be combined to compute the next step during an INFA simulation. Let X_i be k arbitrary subsets of size at most $\lceil n/k \rceil$ partitionning the n states of the INFA $infa(B)$ of the forest B. Each subsets $Y_i \subseteq X_i$, within each subset X_i, are indexed by integers from 0 to $2^{\lceil n/k \rceil}$. Let $\mathbb{D}$ be a function such that $\mathbb{D} : [0, k-1] \times [0, k-1] \times \Sigma \times [0, 2^{\lceil n/k \rceil}] \to [0, 2^{\lceil n/k \rceil}]$. An argument (i, j, λ, x_i) of $\mathbb{D}$ is composed of two values $0 \leq i \leq k-1$ and $0 \leq j \leq k-1$ identifying two subsets X_i and X_j of the partition, a symbol $\lambda \in \Sigma$ and the integer $0 \leq x_i \leq 2^{\lceil n/k \rceil} - 1$ indexing the subset Y_i of X_i. The value $\mathbb{D}(i, j, \lambda, x_i) \in [0, 2^{\lceil n/k \rceil}]$ is an integer indexing a subset Y_j of X_j such that a state s belongs to Y_j iff s belongs to X_j and there is a state in Y_i that transitions to s on input symbol λ. Let a be an abstract tree and b a sequence of k bits b_i. For the initialisation, X_{i_0} is the partition subset containing the initial state of $infa(B)$ and all bits of the sequence b are set to zero excepted the bit whose index corresponds to this initial state. Let $p = 0$ be the position of the symbol $a[p]$ currently read in a. Each integer value $\mathbb{D}(i, j, a[p], b)$ is the index of a subset of the states of the set X_j that can be reached by a transition on $a[p]$ from a state in the set Y_i identified by the sequence b as a subset of the set X_i. Let Z be the set of all integer values $\mathbb{D}(i, j, a[p], b)$ for every pair $i, j \in [0, k-1]$ and let z be the result of the OR bitwise binary operation of all the indices in Z. Now, z is the bit sequence indexing the unique subset Y_j of the set of states X_j reachable in $infa(B)$ by the substring $a[0, p]$ of a. If $p < |a| - 1$ then increment p, set the sequence b equal to z and start again this procedure. Otherwise $p = |a| - 1$: in this case, the string a has been read till the end and the simulation is finished. If Y_j contains at least one accepting states from $infa(B)$ then a is accepted, otherwise it is rejected. The space required is the size for storing the function $\mathbb{D}$ which is at most the cardinal of $[0, k-1] \times [0, k-1] \times \Sigma \times [0, 2^{\lceil n/k \rceil}]$, and thus $O(k^2 \cdot |\Sigma| \cdot 2^{\lceil n/k \rceil})$. For fixed p, $\lambda = a[p]$ and b, each step of the simulation that reads a symbol of a checks the value $\mathbb{D}(i, j, \lambda, b)$ for all $(i, j) \in [0, k-1] \times [0, k-1]$. Thus each step takes time $O(k^2)$, and since there are as many steps as $|a|$ symbols in a then the total simulation time is $O(|a| \cdot k^2)$, which is equal to $O(|A| \cdot k^2)$ where A is the concrete representation of a.

5 Conclusion

The KR language presented in this paper provides concrete and abstract syntaxes for labeled rooted trees defined according to a relational vocabulary Σ. On the one hand, the abstract syntax is provided with a set-theoretic semantics and a subsumption relation matching the existential conjunctive fragment of the description logic $\mathcal{ALC}$. On the other hand, a notion of homomorphism is defined between concrete trees with a linear time algorithm; and shown to be equivalent to subsumption. Moreover, given a tree T with abstract syntax t and a forest

B, there is an homomorphism from T to B: (1) iff T subsumes a tree in B; and (2) iff t belongs to the set L_B of all tours of B. Making use of automata theory, the membership $t \in L_B$ can be checked by running on input string t the automata $infa(B)$ that recognizes the tour language L_B. The first data structure proposed in this paper is obtained by determinizing $infa(B)$ into a deterministic automaton of size at most $O(2^{|B|})$ that reads t in time $O(|t|)$. If t is accepted and K homomorphisms from T to B have to be returned, they can be reconstructed from the run in time $O(K \cdot |T| \cdot \log |B|)$. If the determinization can not be total (e.g. because of space limitations), it is possible to build a second data structure of size $O(k^2 \cdot |\Sigma| \cdot 2^{\lceil |B|/k \rceil})$ that simulates in time $O(k^2 \cdot |t|)$ a run of the automaton $infa(B)$ on input string t; providing a trade-off between time and size depending on the parameter k.

References

1. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 21–43. Springer, Heidelberg (2009). doi:10.1007/978-3-540-92673-3_1
2. Bentley, J.L.: Multidimensional binary search trees used for associative searching. Commun. ACM **18**(9), 509–517 (1975)
3. Bentley, J.L.: Decomposable searching problems. Inf. Process. Lett. **8**(5), 244–251 (1979)
4. Bruno, N., Koudas, N., Srivastava, D.: Holistic twig joins. In: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, SIGMOD 2002. ACM Press (2002)
5. Eppstein, D.: Are there small machines which can efficiently match regular expressions? (Krishnaswami) (2010). https://cstheory.stackexchange.com/questions/1132/are-there-small-machines-which-can-efficiently-match-regular-expressions/1273#1273
6. Eppstein, D., Goodrich, M.T., Sun, J.Z.: Skip quadtrees: dynamic data structures for multidimensional point sets. Int. J. Comput. Geom. Appl. **18**(01n02), 131–160 (2008)
7. Finkel, R.A., Bentley, J.L.: Quad trees a data structure for retrieval on composite keys. Acta Inform. **4**(1), 1–9 (1974)
8. Flouri, T., Janoušek, J., Melichar, B., Iliopoulos, C.S., Pissis, S.P.: Tree template matching in ranked ordered trees by pushdown automata. In: Bouchou-Markhoff, B., Caron, P., Champarnaud, J.-M., Maurel, D. (eds.) CIAA 2011. LNCS, vol. 6807, pp. 273–281. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22256-6_25
9. Gou, G., Chirkova, R.: Efficient algorithms for exact ranked twig-pattern matching over graphs. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008. ACM Press (2008)
10. Knuth, D.E.: Optimum binary search trees. Acta Inform. **1**(1), 14–25 (1971)
11. McClellan, M.T., Minker, J., Knuth, D.E.: The art of computer programming, vol. 3: sorting and searching. Math. Comput. **28**(128), 1175 (1974)
12. Mugnier, M.-L.: On generalization/specialization for conceptual graphs. J. Exp. Theor. Artif. Intell. **7**(3), 325–344 (1995)
13. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. **3**(2), 114–125 (1959)