A Simple Extension to Finite Tree Automata for Defining Sets of Labeled, Connected Graphs

Akio Fujiyoshi[1] and Daniel Průša[2](✉)

[1] Department of Computer and Information Sciences, Ibaraki University, 4-12-1 Nakanarusawa, Hitachi, Ibaraki 316-8511, Japan
akio.fujiyoshi.cs@vc.ibaraki.ac.jp

[2] Faculty of Electrical Engineering, Czech Technical University, Karlovo náměstí 13, 121 35 Prague 2, Czech Republic
prusapa1@fel.cvut.cz

Abstract. This paper introduces spanning tree automata (ST automata) usable for defining sets of labeled, connected graphs. The automata are simply obtained by extending ordinary top-down finite tree automata for labeled, ordered trees. It is shown that ST automata can define any finite set of labeled, connected graphs, and also some subclasses of infinite sets of graphs that can represent the structure of chemical molecules. Although the membership problem for ST automata is NP-complete, an efficient software was developed which supports a practical use of ST automata in chemoinformatics as well as in other fields.

Keywords: Automata theory · Tree automaton · Graph automaton · NP-completeness · Chemoinformatics

1 Introduction

Formal grammars, finite automata and regular expressions are powerful tools for defining sets of strings over a finite alphabet [10,11]. For defining sets of labeled, ordered trees, we have tree grammars and finite tree automata as well [2,4,8,13]. How about for defining sets of graphs? Monadic second-order logic (MSOL) has been studied for describing graph properties [6]. There are many studies of graph grammars [14]. As for practical tools like context-free grammars and regular expressions, however, there is no common idea. This paper suggests the use of finite tree automata with a simple extension for defining sets of graphs.

Simple but powerful tools for defining a set of labeled, connected graphs has been requested in the field of chemoinformatics [3]. Pharmaceutical companies and research laboratories have to claim their intellectual property on chemical

A. Fujiyoshi—Supported by JSPS KAKENHI Grant Number JP18H01036.
D. Průša—Supported by the Czech Science Foundation grant 19-21198S.

structures of new medicine by applying for patents. Chemical structure formulas of molecules are labeled, connected graphs, where vertices are labeled as the name of atoms and edges are labeled as the type of bonds. To protect the intellectual property of new medicine, not only the exact chemical structure of new medicine but also similar chemical structures must be explicitly claimed in a patent application because a chemical compound with a similar chemical structure usually has a similar chemical property. The most popular way for this purpose is using a Markush structure. A Markush structure is a graphical diagram with expressions in a natural language commonly used in patent claims, firstly used in January, 1923. Because of the limitation of its expressive power, the range of a Markush structure often becomes too broad and contains many unrelated chemical compounds [15]. However, a substitutable method has not been invented so far, and Markush structures are continuously used for almost 100 years.

This paper introduces spanning tree automata (ST automata) for defining sets of labeled, connected graphs. ST automata are simply obtained by extending ordinary top-down finite tree automata for labeled, ordered trees. The idea behind this extension is based on a very simple fact: "Any connected graph with cycles becomes a tree if we break all cycles." As shown in Fig. 1, by choosing an edge from each cycle and inserting two virtual vertices at the middle of the edges, a tree is obtained. We call this tree an extended spanning tree. An ST automaton may be viewed as a finite tree automaton that accepts extended spanning trees instead of labeled, connected graphs.

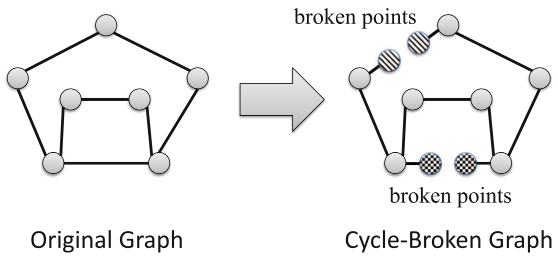

Fig. 1. Obtaining an extended spanning tree.

The goal of the paper is to demonstrate that the proposed ST automata (defined in Sect. 3) are a practical tool which can be applied to describe and analyse structures important for chemoinformatics and other fields. It is shown that ST automata can define any finite set of labeled, connected graphs (Sect. 4). The ability to define infinite sets of graphs is strengthened after introducing a variant of ST automata working over breadth-first search spanning trees only (Sect. 5). Although the membership problem for ST automata is NP-complete, we were able to develop an implementation efficient enough to find subgraphs accepted by an ST automaton in graphs induced by real data. The implementation extends the algorithm in [7] proposed for finding a spanning tree accepted by a finite tree automaton in a graph of treewidth 2. A report on the newly developed software and performed experiments is available in Sect. 6.

2 Preliminaries

A *graph* is an ordered pair $G = (V, E)$, where V is a finite set of *vertices*, and E is a set of unordered pairs of distinct vertices, called *edges*. An edge $\{u, v\} \in E$ is written as uv or vu. A vertex $u \in V$ is *adjacent* to another vertex $v \in V$ if an edge uv is in E. For $v \in V$, we define $N(v) = \{u \mid u \in V \text{ and } u \text{ is adjacent to } v\}$, and $|N(v)|$ is called the *degree* of v.

For vertices $u, v \in V$, a *path* of *length* $n \geq 0$ from u to v is a sequence of vertices $v_1, \ldots, v_{n+1}$ where $u = v_1$, $v = v_{n+1}$, for all $1 \leq i \leq n$, v_i is adjacent to v_{i+1}, and $v_1, v_2, \ldots, v_{n+1}$ are all distinct except that v_1 may equal to v_{n+1}. The *distance* between $u, v \in V$ in G, denoted $d_G(u, v)$, is the length of a shortest path from u to v. A *cycle* is a path of positive length from v to v for some $v \in V$. Graph G is *acyclic* if there is no cycle in G. Graph G is *connected* if there is a path from u to v for any pair of distinct vertices $u, v \in V$.

A *tree* is a connected, acyclic graph. A vertex of a tree is called a *node*. A *rooted tree* is a pair (T, r) such that T is a tree, and r is a node of T. The node r is called the *root*. In a rooted tree, we assume that the edges have a natural direction away from the root. The *parent* of a node v is the node adjacent to v on the path from the root to v. Note that every node except the root has a unique parent. The *children* of a node v are the nodes whose parent is v. A node without any children is called a *leaf*.

Let Σ be a finite set of *vertex labels*, and let Γ be a finite set of *edge labels*. A *vertex labeling* of G is a function $\sigma : V \to \Sigma$, and a *edge labeling* of G is a function $\gamma : E \to \Gamma$. A *labeled* graph over Σ and Γ is a quadruple $G = (V, E, \sigma, \gamma)$. In this paper, we assume every graph to be connected and labeled, and a graph implies a labeled, connected graph unless otherwise stated. We use letters in Roman alphabet $A, a, B, b, C, c, \ldots$ for vertex labels and numerical digits $1, 2, 3, \ldots$ for edge labels.

Let $G = (V, E, \sigma, \gamma)$ be a graph over Σ and Γ, and let $B \subseteq E$ be a subset of edges such that $T = (V, E \setminus B)$ is a tree. B is called a set of *non-tree edges*. An *extended spanning tree* of G decided by B is a tree $T = (V', E', \sigma', \gamma')$ over Σ and Γ defined as follows:

- $V' = V \cup \{b_u, b_v \mid b = uv \in B\}$, where b_u and b_v are new vertices not included in V, called *virtual vertices* for a non-tree edge b.
- $E' = (E \setminus B) \cup \{ub_u, vb_v \mid b = uv \in B\}$.
- $\sigma' : V' \to \Sigma$ is such that, for each $v \in V$, $\sigma'(v) = \sigma(v)$, and, for each $v \in V' \setminus V$, $\sigma'(v)$ is set to undefined.
- $\gamma' : E' \to \Gamma$ is such that, for each $e \in E$, $\gamma'(e) = \gamma(e)$, and, for each $b = uv \in B$, $\gamma'(ub_u) = \gamma'(vb_u) = \gamma(b)$.

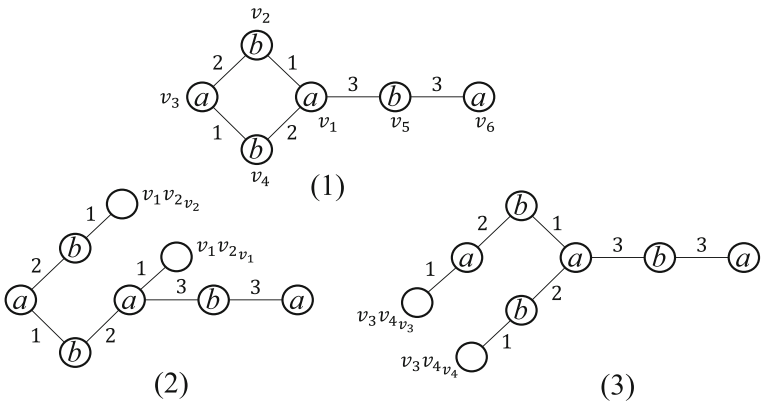

Fig. 2. (1) The graph G, (2) the extended spanning tree decided by B_1, and (3) the extended spanning tree decided by B_2.

Example 1. Let $G = (V, E, \sigma, \gamma)$ be a graph over $\Sigma = \{a, b\}$, $\Delta = \{1, 2, 3\}$, where

$$\begin{aligned}
V =& \{v_1, v_2, v_3, v_4, v_5, v_6\},\\
E =& \{v_1v_2, v_1v_4, v_1v_5, v_2v_3, v_3v_4, v_5v_6\},\\
\sigma =& \{(v_1, a), (v_2, b), (v_3, a), (v_4, b), (v_5, b), (v_6, a)\}, \text{ and}\\
\delta =& \{(v_1v_2, 1), (v_1v_4, 2), (v_1v_5, 3), (v_2v_3, 2), (v_3v_4, 1), (v_5v_6, 3)\}.
\end{aligned}$$

For the graph G, $B_1 = \{v_1v_2\}$ and $B_2 = \{v_3v_4\}$ are two of the sets of non-tree edges. The graph G and the extended spanning trees decided by B_1 and B_2 are illustrated in Fig. 2.

3 Spanning Tree Automata for Labeled, Connected Graphs

A spanning tree automaton is defined as an extension of the well-known nondeterministic top-down finite tree automaton for labeled, ordered trees [4]. Instead of graphs, an ST automaton deals with their extended spanning trees.

3.1 Definitions

A *spanning tree automaton (ST automaton)* is a 6-tuple $A = (Q, \Sigma, \Gamma, q_0, P, R)$ where:

- Q is a finite set of *states*,
- Σ is an alphabet of *vertex labels*,
- Γ is an alphabet of *edge labels*,
- $q_0 \in Q$ is the *initial state*,
- P is a set of unordered pairs of states, called *acceptable state matchings*, and
- R is a finite set of *transition rules* of the form

$$q(f(c_1, c_2, \ldots, c_n)) \to f(q_1(c_1), q_2(c_2), \ldots, q_n(c_n))$$

where $n \geq 0$, $f \in \Sigma$, $q, q_1, q_2, \ldots, q_n \in Q$, and $c_1, c_2, \ldots, c_n \in \Gamma$. The number n is called the *width* of a transition rule. When $n = 0$, we write $q(f) \to f$ instead of $q(f()) \to f()$.

Let $A = (Q, \Sigma, \Gamma, q_0, P, R)$ be an ST automaton, let $G = (V, E, \sigma, \gamma)$ be a graph, let $T = (V', E', \sigma', \gamma')$ be an extended spanning tree of G decided by a set of non-tree edges B, and let $r \in V$ be a vertex of G. A *state mapping* on T is a function $\mu : V' \to Q$. A state mapping μ on the rooted tree (T, r) is *acceptable* by A if the following conditions hold:

– $\mu(r) = q_0$, i.e., a state mapped to the root is always the initial state,
– for each node $v \in V'$ with n $(n > 0)$ children $v_1, v_2, \ldots, v_n$, if $\sigma(v) = f$, $\mu(v) = q$, $\gamma(vv_1) = c_1$, $\gamma(vv_2) = c_2$, $\ldots$, $\gamma(vv_n) = c_n$, and $\mu(v_1) = q_1$, $\mu(v_2) = q_2$, $\ldots$, $\mu(v_n) = q_n$, then R contains the following transition rule:

$$q(f(c_1, c_2, \ldots, c_n)) \to f(q_1(c_1), q_2(c_2), \ldots, q_n(c_n)),$$

– for each leaf $v \in V$, if $\sigma(v) = f$ and $\mu(v) = q$, then R contains the following transition rule:

$$q(f) \to f,$$

– and for each $b = uv \in B$, $\{\mu(b_u), \mu(b_v)\} \in P$, i.e., the states mapped to the virtual vertices for b must be in acceptable state matchings.

The graph G is *accepted* by A if an extended spanning tree T decided by some set of non-tree edges B exists, a state mapping μ on T exists, a vertex $r \in V$ exists, and μ on (T, r) is acceptable by A.

The set of graphs *defined* by an ST automaton A is the set accepted by A.

Example 2. $A = (Q, \Sigma, \Gamma, q_0, P, R)$ is an example of an ST automaton, where $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $\Gamma = \{1, 2, 3\}$, $P = \{\{q_1, q_1\}, \{q_1, q_2\}\}$, and R consists of transition rules:

$$q_0(a(1,2,3)) \to a(q_1(1), q_2(2), q_3(3))\,, \quad q_1(a(2)) \to a(q_2(2))\,,$$
$$q_2(b(1)) \to b(q_1(1))\,, \quad q_3(b(3)) \to b(q_3(3))\,, \quad q_3(a) \to a\,.$$

Consider the graph G and its non-tree edge set B_1 in Example 1. Let G' be the extended spanning tree of G decided by B_1. Consider the following state mapping μ on G':

$$\mu = \{(v_1, q_0), (v_2, q_2), (v_3, q_1), (v_4, q_2), (v_5, q_3), (v_6, q_3), (v_1v_{2v_1}, q_1), (v_1v_{2v_2}, q_1)\}\,.$$

The state mapping μ on (G', v_1) is illustrated in Fig. 3. The graph G is accepted by A because μ on (G', v_1) is acceptable by A. Note that the pair of states mapped to the virtual vertices for v_1v_2 is $\{q_1, q_1\}$, and it is in P.

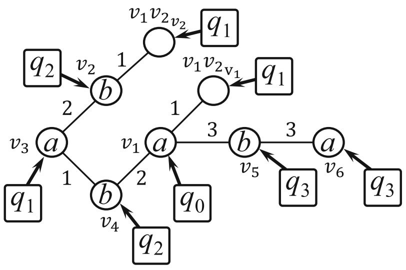

Fig. 3. The state mapping μ on (G', v_1).

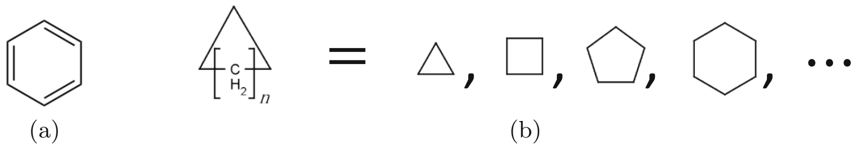

Fig. 4. Chemical structural formulas of (a) benzene and (b) cycloalkanes.

3.2 Examples of Spanning Tree Automata

Example 3. We present two ST automata defining a set of chemical structural formulas. One defines the chemical structural formula of benzene as illustrated in Fig. 4a, and the other defines the cycloalkanes as illustrated in Fig. 4b. In a chemical structural formula, carbon atoms are implied to be located at the vertices of line segments, and hydrogen atoms attached to carbon atoms are omitted.

We set $\Sigma = \{C\}$ and $\Gamma = \{1, 2\}$, where the vertex label C stands for a carbon atom, and the edge labels 1 and 2 stand for a single bond and a double bond, respectively.

The following is an ST automaton that defines a chemical structural formula of benzene: $A = (Q, \Sigma, \Gamma, q_0, P, R)$, where $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$, $P = \{\{q_6, q_6\}\}$ and R consists of transition rules:

$$q_0(C(1,2)) \to C(q_1(1), q_6(2))\,, \quad q_1(C(2)) \to C(q_2(2))\,, \quad q_2(C(1)) \to C(q_3(1))\,,$$
$$q_3(C(2)) \to C(q_4(2))\,, \quad q_4(C(1)) \to C(q_5(1))\,, \quad q_5(C(2)) \to C(q_6(2))\,.$$

The following is an ST automaton that defines chemical structural formulae of the cycloalkanes: $A = (Q, \Sigma, \Gamma, q_0, P, R)$, where $Q = \{q_0, q_1, q_2, q_3\}$, $P = \{\{q_3, q_3\}\}$ and R consists of transition rules:

$$q_0(C(1,1)) \to C(q_1(1), q_3(1))\,, \quad q_1(C(1)) \to C(q_2(1))\,,$$
$$q_2(C(1)) \to C(q_2(1))\,, \quad q_2(C(1)) \to C(q_3(1))\,.$$

4 Properties of Spanning Tree Automata

Lemma 1. *For any graph G there is an ST automaton A defining the set $\{G\}$.*

Proof. Suppose that $G = (V, E, \sigma, \gamma)$, a graph over Σ and Γ, is given. Let B be a set of non-tree edges of G, and let $G' = (V', E', \sigma', \gamma')$ be the extended spanning tree of G decided by B. Choose any vertex $r \in V$ for the root.

We can construct an ST automaton $A = (Q, \Sigma, \Gamma, q_0, P, R)$ as follows:

- $Q = \{\hat{v} \mid v \in V\} \cup \{\hat{p} \mid p \in B\}$,
- $q_0 = \hat{r}$,
- $P = \{\{\hat{p}, \hat{p}\} \mid p \in B\}$, and
- for each $v \in V$, R has the following transition rule:

$$\hat{v}(f(c_1, c_2, \ldots, c_n)) \rightarrow f(q_1(c_1), q_2(c_2), \ldots, q_n(c_n))$$

  where v has n children $v_1, v_2, \ldots, v_n$ in the rooted tree (G', r), $f = \sigma(v)$, and, for $1 \leq i \leq n$, if $v_i = p_v$ for some $p = uv \in B$, then $c_i = \gamma(uv)$ and $q_i = \hat{p}$, otherwise $c_i = \gamma(vv_i)$ and $q_i = \hat{v}_i$.

An acceptable state mapping μ on (G', r) can be obtained as follows: For $v \in V$, $\mu(v) = \hat{v}$, and, for $p = uv \in B$, $\mu(p_u) = \mu(p_v) = \hat{p}$.

Clearly, the ST automaton A precisely defines the set $\{G\}$. □

Lemma 2. *The class of sets of graphs accepted by ST automata is effectively closed under union.*

Proof. Let $A = (Q, \Sigma, \Gamma, q_0, P, R)$ and $A' = (Q', \Sigma', \Gamma', q_0', P', R')$ be ST automata. We may assume that $Q \cap Q' = \emptyset$. We consider an ST automaton $A'' = (Q'', \Sigma'', \Gamma'', q_0'', P'', R'')$ defined as follows:

- $Q'' = Q \cup Q' \cup \{q_0''\}$, where q_0'' is a new state,
- $\Sigma'' = \Sigma \cup \Sigma'$,
- $\Gamma'' = \Gamma \cup \Gamma'$,
- $P'' = P \cup P'$, and
- $R'' = R \cup R' \cup \{r'' \mid r \in R \cup R'$, q_0 or q_0' appears on the left-hand side of r, and r'' is obtained by replacing q_0 or q_0' on the left-hand side of r with $q_0''\}$.

Clearly, a graph G is accepted by A'' if and only if G is accepted by either A or A'. □

Theorem 1. *For any finite set of graphs, we can construct an ST automaton that defines it.*

Proof. The theorem clearly follows from Lemmas 1 and 2. □

Because the membership problem for ST automata includes the graph isomorphism problem, it is at least GI-hard. Its NP-completeness is obtained by the reduction from the Hamiltonian cycle problem with vertex degree at most 3 [9]. When a graph with degree at most 3 has a Hamiltonian cycle, (1) the graph has no vertex of degree 0 or 1, (2) for each vertex of degree 2, both edges connected to the vertex belong to the Hamiltonian cycle, and (3) for each vertex of degree 3, two edges connected to the vertex belong to the Hamiltonian cycle but the remaining edge does not and is adjacent to another vertex of degree 3.

Theorem 2. *The membership problem for ST automata is NP-complete.*

Proof. Consider the ST automaton $A = (Q, \Sigma, \Gamma, q_0, P, R)$, where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a\}$, $\Gamma = \{1\}$, $P = \{\{q_1, q_1\}, \{q_2, q_2\}\}$ and R consists of transition rules:

$$q_0(a(1,1)) \rightarrow a(q_1(1), q_1(1)), \quad q_0(a(1,1,1)) \rightarrow a(q_1(1), q_1(1), q_2(1)),$$
$$q_1(a(1)) \rightarrow a(q_1(1)), \quad q_1(a(1,1)) \rightarrow a(q_1(1), q_2(1)).$$

It is clear that A accepts a graph with degree at most 3 that has a Hamiltonian cycle. Since the Hamiltonian cycle problem with vertex degree at most 3 is NP-hard, this problem is also NP-hard.

On the other hand, given a graph G, we can nondeterministically obtain a set of non-tree edges B, an extended spanning tree G' decided by B, a state mapping μ on G', a vertex $r \in V$, and check if μ on (G', r) is acceptable by A in polynomial time. Thus the problem is in the class NP. □

5 Breadth-First Search Spanning Tree Automata

ST automata can define some infinite sets of graphs, like the set of chemical structural formulas of cycloalkanes shown in Example 3. On the other hand, it is not difficult to find infinite sets of graphs impossible for any ST automaton to define. For example, there is no ST automaton accepting any set of graphs where the graph maximum degree is not bounded. This might not seem too restrictive as vertex degrees of chemical structures are usually bounded. However, there is a more limiting restriction which comes from the observation that if an ST automaton A accepts a graph whose number of minimum cycles is greater than the number of acceptable state matchings of A, then some pairs of virtual vertices are mapped in the same state matching. Consequences are demonstrated by the following example.

Example 4. The set of chemical structural formulas for benzene and acenes, shown in Fig. 5, cannot be defined by any ST automaton.

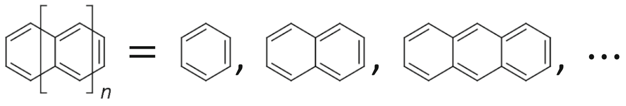

Fig. 5. Chemical structural formulas for benzene and acenes.

There is no limit for the number of minimum cycles of the defined graphs. From this reason, for any ST automaton A, there is a graph G in the defined set such that its number of minimum cycles exceeds the number of acceptable state matchings of the automaton. If two pairs of virtual vertices (uv_u, uv_v) and (st_s, st_t) are mapped to the same state matching, and u, v, s, t are all distinct, then another graph obtained from G by removing the edges uv and st and adding the new edges ut and vs is accepted by A. However, the newly obtained graph is not a chemical structural formula for an acene anymore.

To handle chained structures like acenes by ST automata, we introduce a variant of the automata working over extensions of breadth-first search (BFS, [5]) spanning trees only. A breadth-first search spanning tree automaton (BFS-ST automaton) is a six-tuple $A = (Q, \Sigma, \Gamma, q_0, P, R)$ where the components have the same meaning as in the case of the ST automaton defined in Subsect. 3.1.

A graph $G = (V, E, \delta, \gamma)$ is accepted by A if G has a BFS spanning tree $T = (V, E_T)$ rooted in a vertex r (i.e., for all $v \in V$, it holds that $d_G(r, v) = d_T(r, v)$) such that the pair (T', r), where T' is the extended spanning tree decided by $E \setminus E_T$, is accepted by A for some state mapping μ.

We give a characterization of a certain family of infinite sets of graphs accepted by BFS-ST automata, which includes the set of graphs representing benzene and acenes. The idea behind the family is (1) to take a word w from a regular language, (2) to substitute graphs for symbols of w (where occurrences of the same symbol are substituted by isomorphic graphs), and (3) to combine neighboring graphs by merging some of their predefined pairs of vertices. A detailed description of this construction follows.

Let Σ and Γ be finite sets of vertex and edge labels, respectively. We say that $C = (V, E, \sigma, \gamma, \boldsymbol{V}^+, \boldsymbol{V}^-)$ is a *graph component* over (Σ, Γ) if $C' = (V, E, \sigma, \gamma)$ is a labeled, connected graph over Σ, and Γ, $\boldsymbol{V}^+$ and $\boldsymbol{V}^-$ are non-empty vectors of distinct elements of V, there is no edge in E connecting a pair of vertices from $\boldsymbol{V}^-$, and there is an integer $d \geq 1$ such that $\forall u \in \boldsymbol{V}^+, \forall v \in \boldsymbol{V}^- : d'_C(u, v) \in \{d, d+1\}$. An example of graph components is given in Fig. 6.

Let L be a regular language over an alphabet Σ_1. Let $\mathrm{FIRST}(L) \subseteq \Sigma_1$ be the set of symbols that begin a word from L, and $\mathrm{FOLLOW}(L, a) \subseteq \Sigma_1$ be the set of symbols $b \in \Sigma_1$ for which there is a word in L in which b follows after a.

A *component substitution function* π is a function assigning to each symbol of Σ_1 a graph component over (Σ, Γ). Let $\pi(a) = C_a = (V_a, E_a, \sigma_a, \gamma_a, \boldsymbol{V}_a^+, \boldsymbol{V}_a^-)$ for each $a \in \Sigma_1$. We say π is *compatible* with L iff for all $a \in \Sigma_1$, $b \in \mathrm{FOLLOW}(L, a)$, it holds that $|\boldsymbol{V}_a^-| = |\boldsymbol{V}_b^+|$, and, for all $a \in \mathrm{FIRST}(L)$, it holds that $|\boldsymbol{V}_a^+| = 1$. Note that $|\boldsymbol{v}|$ denotes the dimension of a vector $\boldsymbol{v}$.

For a π compatible with L, we define $\mathcal{G}(L, \pi)$ to be the set of graphs $G(w, \pi)$ where $w = a_1 \dots a_n$, with $a_i \in \Sigma_1$, is a non-empty word from L and the graph $G(w, \pi) = (V, E, \sigma, \gamma)$ is constructed as follows. Let $C_i = (V_i, E_i, \sigma_i, \gamma_i, \boldsymbol{V}_i^+, \boldsymbol{V}_i^-)$ be a graph component isomorphic to $\pi(a_i)$. Assume that $V_i \cap V_j = \emptyset$ for all $1 \leq i < j \leq n$. Treat vectors of distinct elements as sets and define a mapping

$$\nu : \bigcup_{i=1}^{n} V_i \to \bigcup_{i=1}^{n} V_i \setminus \bigcup_{i=1}^{n-1} \boldsymbol{V}_i^-$$

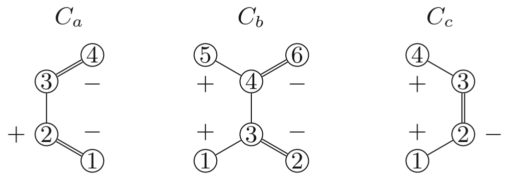

Fig. 6. Graph components $C_s = (V_s, E_s, \sigma_s, \gamma_s, \boldsymbol{V}_s^+, \boldsymbol{V}_s^-)$, $s \in \{a, b, c\}$, where it holds that $\boldsymbol{V}_a^+ = (2)$, $\boldsymbol{V}_a^- = (1,4)$, $\boldsymbol{V}_b^+ = (1,5)$, $\boldsymbol{V}_b^- = (2,6)$, $\boldsymbol{V}_c^+ = (1,4)$ and $\boldsymbol{V}_c^- = (2)$. Vertices of the sets $\boldsymbol{V}_s^+$ and $\boldsymbol{V}_s^-$ are distinguished by symbols + and −, respectively. We define $\gamma_s : E_s \to \{0,1\}$ where labels 0 and 1 represent single and double bonds, respectively.

fulfilling: if v is the j-th component of $\boldsymbol{V}_i^-$, where $i < n$, then $\nu(v)$ equals the j-th component of $\boldsymbol{V}_{i+1}^+$, otherwise $\nu(v) = v$. Then, $V = \bigcup_{i=1}^n V_i \setminus \bigcup_{i=1}^{n-1} \boldsymbol{V}_i^- = \{\nu(v) \mid v \in \bigcup_{i=1}^n V_i\}$, $E = \{\{\nu(u), \nu(v)\} \mid \{u,v\} \in \bigcup_{i=1}^n E_i\}$, and $\sigma(\nu(v)) = \sigma_i(v)$, $\gamma(\{\nu(u), \nu(v)\}) = \gamma_i(\{u,v\})$ for all $v \in V_i$, $\{u,v\} \in E_i$, $i = 1, \ldots, n$.

Example 5. Let L be a regular language over $\Sigma_1 = \{a, b, c\}$ defined by the regular expression ab^*c. Let $\pi : \Sigma_1 \to \{C_a, C_b, C_c\}$ be a component substitution function such that $\pi(a) = C_a$, $\pi(b) = C_b$, $\pi(c) = C_c$ where the graph components are those shown in Fig. 6. Observe that π is compatible with L. Then, $\mathcal{G}(L, \pi)$ is a set of graphs that represent benzene and acenes.

Theorem 3. *For any regular language L over Σ_1 and any component substitution function $\pi : \Sigma_1 \to \mathcal{C}$, where $\mathcal{C}$ is a set of graph components over (Σ, Γ) and π is compatible with L, there is a BFS-ST automaton A accepting $\mathcal{G}(L, \pi)$.*

We omit the proof of this theorem because of lack of space.

6 Implementation and Experiments

For practical use of ST automata, a search software which for a given dataset of graphs finds subgraphs accepted by an ST automaton was developed. It allows to search for acceptable subgraphs of the maximum or minimum size. The source code of the software, written in the C++ programming language, is available on the web site: http://apricot.cis.ibaraki.ac.jp/CBGfinder/.

The graphset-subgraph matching algorithm used in the software is an extension of the algorithm described in [7], which was designed to find a spanning tree accepted by a finite tree automaton in an input graph of treewidth 2. The new algorithm was developed based on the original one with the following extensions: (1) It searches for graphs defined by ST or BFS-ST automata instead of spanning trees defined by tree automata. (2) It is optionally able to search for subgraphs and induced subgraphs (defined by an ST or BFS-ST automaton) in a given input graph. (3) There is no limit on the input graph treewidth.

The algorithm is dynamic programming based and works in a bottom-up manner to construct an acceptable state mapping. Partial solutions established

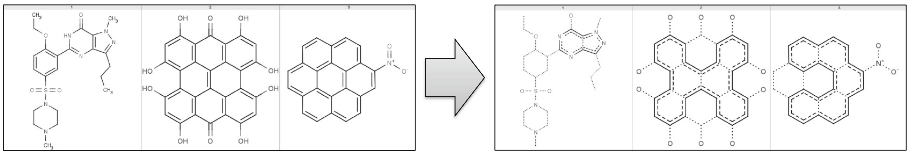

Fig. 7. Longest aromatic cycles detected by the search software. Three samples of input molecules are shown on the left. All parts of the found longest aromatic cycles are highlighted by the dashed line segments on the right.

over subgraphs are combined to reach a global solution. For a graph $G = (V, E)$ and an ST automaton A, the maximum number of partial solutions over a subgraph is $T = (r \cdot 2^{width} \cdot tw)^{tw}$, where tw is the treewidth of G, r is the number of transition rules of A, and $width$ is the maximum width of a transition rule. It can be derived that the space complexity is $O(|E| \cdot T)$, and the time complexity is $O(|E| \cdot T^2 \cdot \log(T))$. Note that these estimates are theoretical upper bounds, the actual space consumptions and running times are typically lower due to pruning many of the partial solutions that do not extend to larger ones.

To demonstrate a practical usage of the software, an evaluation was conducted to find the longest aromatic cycle of each molecule stored in the ChEMBL database. ChEMBL is a database of bioactive drug-like small molecules maintained by the European Bioinformatics Institute. 635,933 molecules are stored in ChEMBL version 8. Among them, 87 molecules containing a fullerene structure are excepted because their treewidth is too big. The number of atoms (bonds) of a molecule varies from 1 (0) to 878 (895) and is 32.02 (34.46) on average. An ST automaton defining the set of aromatic cycles was used.

As a result, among 635,846 molecules, 580,354 molecules were accepted with largest aromatic cycles detected, and 55,492 molecules without aromatic cycles were rejected. The evaluation was finished in 353.4 s in total (0.55 ms per item on average). The molecule which took the longest time (0.028 s) was the rightmost one in Fig. 7. The specification of the machine used for the experiment is as follows: Intel core i5-5200U (2.20GHz) CPU, 8 GB RAM, and Microsoft Windows 7 (64Bit) OS.

The software can be used not only for chemoinformatics but also for various NP-hard graph problems such as subgraph isomorphism, travelling salesman, longest path, feedback vertex set, Steiner tree, and so on.

To demonstrate this, we briefly report on the Third Parameterized Algorithms and Computational Experiments Challenge (PACE 2018) competition [12], in which the software participated in Track B. The task was to compute an optimal Steiner tree of a given graph within a given time limit on the same public environment. For this competition, a well-known speed-up technique [1] was implemented. The software correctly solved 49 out of 100 competition instances, despite it is not fully optimized for this particular task (for comparison, the best five participating systems solved 92, 77, 58, 52 and 52 instances, respectively).

7 Conclusions

We introduced ST automata defining sets of labeled, connected graphs. We demonstrated that they are suitable for detection of chemical structures in molecules, which was confirmed by experiments. As a future work, regular expressions for the automata should be considered. It would also be beneficial to identify broader subclasses of sets of graphs defined by BFS-ST automata.

References

1. Bodlaender, H.L., Cygan, M., Kratsch, S., Nederlof, J.: Deterministic single exponential time algorithms for connectivity problems parameterized by treewidth. Inf. Comput. **243**, 86–111 (2015). https://doi.org/10.1016/j.ic.2014.12.008
2. Brainerd, W.S.: Tree generating regular systems. Inf. Control **14**(2), 217–231 (1969). https://doi.org/10.1016/S0019-9958(69)90065-5
3. Brown, N.: Chemoinformatics-an introduction for computer scientists. ACM Comput. Surv. **41**(2), 8:1–8:38 (2009). https://doi.org/10.1145/1459352.1459353
4. Comon, H., et al.: Tree automata techniques and applications (2007). http://tata.gforge.inria.fr/. Accessed 12 Oct 2007
5. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009). http://mitpress.mit.edu/books/introduction-algorithms
6. Courcelle, B., Engelfriet, J.: Graph Structure and Monadic Second-Order Logic - A Language-Theoretic Approach, Encyclopedia of Mathematics and its Applications, vol. 138. Cambridge University Press, Cambridge (2012)
7. Fujiyoshi, A.: A practical algorithm for the uniform membership problem of labeled multidigraphs of tree-width 2 for spanning tree automata. Int. J. Found. Comput. Sci. **28**(5), 563–582 (2017). https://doi.org/10.1142/S012905411740007X
8. Fujiyoshi, A., Kasai, T.: Spinal-formed context-free tree grammars. Theory Comput. Syst. **33**(1), 59–83 (2000). https://doi.org/10.1007/s002249910004
9. Garey, M., Johnson, D., Stockmeyer, L.: Some simplified NP-complete graph problems. Theoret. Comput. Sci. **1**(3), 237–267 (1976). https://doi.org/10.1016/0304-3975(76)90059-1
10. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
11. Kleene, S.C.: Representation of events in nerve nets and finite automata. In: Shannon, C.E., McCarthy, J. (eds.) Automata Studies, pp. 3–42. Princeton University Press, Princeton (1951)
12. PACE 2018. https://pacechallenge.org/2018/steiner-tree/
13. Rounds, W.C.: Mapping and grammars on trees. Math. Syst. Theory **4**(3), 257–287 (1970). https://doi.org/10.1007/BF01695769
14. Rozenberg, G., Ehrig, H., Engels, G., Kreowski, H., Montanari, U. (eds.): Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1–3. World Scientific (1997–1999)
15. Sibley, J.F.: Too broad generic disclosures: a problem for all. J. Chem. Inf. Comput. Sci. **31**(1), 5–9 (1991). https://doi.org/10.1021/ci00001a002