

Active Learning for Deterministic Bottom-up Nominal Tree Automata

Rindo Nakanishi¹, Yoshiaki Takata², and Hiroyuki Seki¹

¹ Graduate School of Informatics, Nagoya University
Furo-cho, Chikusa, Nagoya 464-8601, Japan
`rindo@sqlab.jp`, `seki@i.nagoya-u.ac.jp`

² Graduate School of Engineering, Kochi University of Technology
Tosayamada, Kami City, Kochi 782-8502, Japan
`takata.yoshiaki@kochi-tech.ac.jp`

Abstract. Nominal set plays a central role in a group-theoretic extension of finite automata to those over an infinite set of data values. Moerman et al. proposed an active learning algorithm for nominal word automata with the equality symmetry. In this paper, we introduce deterministic bottom-up nominal tree automata (DBNTA), which operate on trees whose nodes are labelled with elements of an orbit finite nominal set. We then prove a Myhill-Nerode theorem for the class of languages recognized by DBNTA and propose an active learning algorithm for DBNTA. The algorithm can deal with any data symmetry that admits least support, not restricted to the equality symmetry and/or the total order symmetry. To prove the termination of the algorithm, we define a partial order on nominal sets and show that there is no infinite chain of orbit finite nominal sets with respect to this partial order between any two orbit finite sets.

Keywords: nominal tree automata, active learning, Myhill-Nerode theorem

1 Introduction

Computational models such as finite automaton, pushdown automaton and context-free grammar provide a theoretical basis for automated technologies including model checking, testing and synthesis. Although the technologies have brought fruitful success, these models cannot directly deal with data values. However, if we add to a classical model the ability of processing data values, the resulting model easily becomes Turing machine-equivalent and the decidability needed for automated technologies is lost. Register automaton (RA) is an extension of finite automaton (FA) by adding registers for manipulating data values in a restricted way [12]. RA can compare an input data value with those stored in its registers to determine its behavior. RA inherits some of the good properties from FA including closure properties on language operations and the decidability of basic problems. For example, the membership and emptiness problems are decidable for RA and their complexities are extensively studied [12,22,19,15]. Similar

extensions of other classical models have been proposed such as register tree automaton [13,10], register context-free grammar [6,23] and register pushdown automaton [18,24]. Logics on data words have also been proposed including LTL with the freeze quantifier [9] and two-variable first-order logic [2].

A common property of these models is that the behavior of an automaton (or a grammar) does not depend on data values themselves, but on the relationship (e.g., equality, total order) among data values. Assume that the comparison operator of RA is only equality check. Also assume that an RA A has one register to store the first data value of an input word and test whether the remaining data values are different from the data value in the register except the last one in the input, which should be the same as the first data value. Then, A accepts data words $2 \cdot 5 \cdot 6 \cdot 2$, $8 \cdot 1 \cdot 3 \cdot 8$, and so on. Note that the data word $8 \cdot 1 \cdot 3 \cdot 8$ can be obtained from $2 \cdot 5 \cdot 6 \cdot 2$ by the permutation that maps 2, 5 and 6 to 8, 1 and 3, respectively.

The above observation gives us a group-theoretic extension of FA [3]. Assume that we are given a countable set \mathbb{D} of data values and a permutation group G on \mathbb{D} . To deal with data values in a restricted way, we use orbit finite sets instead of finite sets to represent both of an alphabet and a set of states. A set X is called a G -set if X is equipped with actions (or operations) having the group structure G . Let X be a G -set. The orbit of $x \in X$ is the set $\{x \cdot \pi \mid \pi \in G\}$. X is *orbit finite* if X is divided into a finite number of orbits. A set $C \subseteq \mathbb{D}$ is a *support* of $x \in X$ if any action π that acts as identity on C does not move x , i.e., maps x to x . X is *nominal* if every $x \in X$ has a finite support. Intuitively, X is nominal if for every $x \in X$, all the information on x can be represented by a finite subset of data values, which corresponds to the contents of registers. A nominal automaton over an orbit finite alphabet consists of an orbit finite nominal set of states and an (equivariant) transition relation on states.

Automated learning methods are incorporated into software verification and testing (see [14,7] for an overview). Two well-known applications are black-box checking [20] and compositional verification [8]. Among others, Angluin's L^* algorithm [1] is frequently used in these methods. The algorithm learns the minimum FA for an unknown regular language U by constructing an observation table. Rows and columns of the table are sample input strings and each entry of the table is 1 (accept) or 0 (reject). The algorithm expands the table based on answers from a teacher (oracle) of U for membership and equivalence queries until the teacher answers yes to an equivalence query. The correctness of L^* is guaranteed by the Myhill-Nerode theorem for regular languages. The L^* algorithm has been extended for register automata (e.g. [4,5]) and a learning tool RALib is implemented [5]. In RA, a state transition depends on the comparison among an input data value and those stored in the registers specified as the guard condition of an applied transition rule. Due to this feature of RA, an entry of an observation table is not just 0/1 but more complex information that represents the guard condition of a transition in RA (a symbolic decision tree in [5]), which makes the algorithm rather complicated. Moerman et al. proposed an L^* -style algorithm for nominal word automata [17]. Their algorithm recovers

the simplicity of the original L^* algorithm by the abstract feature of nominal automaton, which is independent of a concrete representation of an automaton. However, the algorithm assumes the equality symmetry as the structure of the set of data values. Moreover, tree models that can manipulate data values are needed for the basis of XML document processing because an XML document usually contains data values associated with structural information represented by a tree [16,15]. For such applications, tree automata theory based on nominal sets should be developed.

In this paper, we define deterministic bottom-up nominal tree automata (DBNTA), which operate on trees whose nodes are labelled with elements of an orbit finite nominal set. We then prove a Myhill-Nerode theorem for the class of languages recognized by DBNTA and propose an active learning algorithm for DBNTA based on the theorem. **The algorithm can deal with any data symmetry that admits least support, not restricted to the equality symmetry and/or the total order symmetry.** To prove the termination of the algorithm, we define a partial order on nominal sets and show that there is no infinite chain of orbit finite nominal sets with respect to this partial order between any two orbit finite sets.

2 Preliminaries

2.1 Nominal set

Let G be a group and X be a set. A group action of G on X is a function $\cdot : X \times G \rightarrow X$ satisfying

$$x \cdot e = x \quad \text{and} \quad x \cdot (\pi\sigma) = (x \cdot \pi) \cdot \sigma$$

for all $x \in X$ and $\pi, \sigma \in G$, where $e \in G$ is the neutral element of G and $\pi\sigma$ is the product of π and σ on G . We call a set with a group action of G a G -set.

We define the *orbit* of $x \in X$ as $Orbit(x) = \{x \cdot \pi \mid \pi \in G\} \subseteq X$. A G -set is uniquely partitioned into different orbits. A G -set consisting of one orbit is called a *single orbit* set, and a G -set consisting of a finite number of orbits is called an *orbit finite* set. We define an *alphabet* as an orbit finite set.

Let X be a G -set. $Y \subseteq X$ is called *equivariant* if $y \in Y \Rightarrow y \cdot \pi \in Y$ holds for all $\pi \in G$. Equivalently, this means that Y is a union of some orbits of X . In the same way, for G -sets X and Y , a binary relation $R \subseteq X \times Y$ is called equivariant if $(x, y) \in R \Rightarrow (x \cdot \pi, y \cdot \pi) \in R$ holds for all $\pi \in G$. An n -ary equivariant relation is defined in the same way for $n \geq 3$. If a binary relation $f \subseteq X \times Y$ is a function, f is equivariant if and only if $f(x \cdot \pi) = f(x) \cdot \pi$ holds for all $x \in X$ and $\pi \in G$.

Let \mathbb{D} be a countable set of data values and G be a permutation group of \mathbb{D} , i.e., a subgroup of the symmetric group $\mathbf{Sym}(\mathbb{D})$ of \mathbb{D} . We call (\mathbb{D}, G) a *data symmetry*. We show some examples of data symmetries. The equality symmetry is $(\mathbb{N}, \mathbf{Sym}(\mathbb{N}))$, where \mathbb{N} is the set of natural numbers and $\mathbf{Sym}(\mathbb{N})$ is the group of all bijections on \mathbb{N} . The total order symmetry is $(\mathbb{Q}, G_{<})$ where \mathbb{Q} is the set of rational numbers and $G_{<}$ is the group of monotone bijections on \mathbb{Q} . The

integer symmetry is $(\mathbb{Z}, G_{\mathbb{Z}})$ where \mathbb{Z} is the set of integers and $G_{\mathbb{Z}}$ is the group of translations $i \mapsto i + c$ for $c \in \mathbb{Z}$.

Let $x \in X$ and $C \subseteq \mathbb{D}$. If for every $\pi \in G$,

$$(\forall c \in C. \pi(c) = c) \Rightarrow x \cdot \pi = x$$

holds, we say that C *supports* x or C is a *support* of x . That is, C supports x if every π which is the identity on C does not move x . A G -set is *nominal*, if every element of the set has a finite support. In any data symmetry (\mathbb{D}, G) , \mathbb{D} itself is a nominal G -set because any element $d \in \mathbb{D}$ has a support $\{d\} \subseteq \mathbb{D}$. \mathbb{D}^* is also nominal because any element $d_1 d_2 \cdots d_n \in \mathbb{D}^*$ has a support $\{d_1, d_2, \dots, d_n\}$. On the other hand, \mathbb{D}^ω , the set of infinite sequences over \mathbb{D} , is not nominal. In the following, we just call an alphabet that is nominal a *nominal alphabet*.

Let $C \subseteq \mathbb{D}$ be a support of $x \in X$. If all supports of x are supersets of C , C is the *least support* of x . For a data symmetry (\mathbb{D}, G) , if every element of every nominal G -set has a least support, the data symmetry *admits least support*.

It is shown in [11] that the equality symmetry and the total order symmetry admit least supports. (Also see [3, Corollaries 9.4 and 9.5].) In the integer symmetry, every $G_{\mathbb{Z}}$ -set is nominal by the following reason. If a translation $i \mapsto i + c$ on \mathbb{Z} does not move an integer $z \in \mathbb{Z}$, the translation must be the identity. Hence, any element x of any $G_{\mathbb{Z}}$ -set is supported by a singleton set of an arbitrary integer. For the same reason, the integer symmetry does not admit least support.

For a function $f : X \rightarrow Y$ and a subset $Z \subseteq X$, we write the function whose domain is restricted to Z as $f|_Z$. For functions $f : X \rightarrow Y$ and $g : Y' \rightarrow Z$, we define $fg : X' \rightarrow Z$ as $fg(x) = g(f(x))$ where $X' = \{x \in X \mid f(x) \in Y'\}$.

Let (\mathbb{D}, G) be a data symmetry that admits least support, $C \subseteq \mathbb{D}$ be a finite and fungible³ set and $S \leq \text{Sym}(C)$ be a permutation group on C . For injective functions u and v from C to \mathbb{D} that extend to permutations from G (i.e., $u = \pi|_C$ and $v = \sigma|_C$ for some $\pi, \sigma \in G$), we define $u \equiv_S v$ if and only if $uv^{-1} \in S$ (which equivalently means $\tau u = v$ for some $\tau \in S$). It is easy to see that \equiv_S is an equivalent relation, and thus \equiv_S divides the set of all injections from C to \mathbb{D} that extend to permutations from G into equivalent classes. The equivalent class of u defined by \equiv_S is written as $[u]_S$. For these C and S , the G -set $\llbracket C, S \rrbracket$ is defined as the set of all equivalent classes defined by \equiv_S , i.e. $\llbracket C, S \rrbracket = \{[\pi|_C]_S \mid \pi \in G\}$, where the G -action is defined as $[u]_S \cdot \pi = [u\pi]_S$ for all $\pi \in G$. S is called a local symmetry. As we noted before, C corresponds to the set of (canonical) data values in the registers. By definition, an element of $\llbracket C, S \rrbracket$ is an equivalent class defined by \equiv_S of an injection from C to \mathbb{D} that extends to some permutation in G . As shown in the example in the introduction, an automaton cannot distinguish between C and the set of data values C' obtained from C by any injection from C to \mathbb{D} which is consistent with G . Such an injection

³ A finite set $C \subseteq \mathbb{D}$ is *fungible* if for every $c \in C$ there exists a $\pi \in G$ such that $\pi(c) \neq c$ and $\pi(c') = c'$ for all $c' \in C \setminus \{c\}$. Fungibility is a technical condition guaranteeing that $\llbracket C, S \rrbracket$ is a single orbit nominal set, but it is not directly related to the paper.

$u : C \rightarrow \mathbb{D}$ represents this change of data values in the registers from C to C' , which is indistinguishable from the automaton.

Next, we describe an intuitive meaning of S . For example, if S consists of only the identity, it means that the order of registers is relevant. If $C = \{1, 2, 3\}$ and $S = \{id, a\}$ where id is the identity and a swaps 1 and 2, then S means that the order between the first and second values are irrelevant. Note that a standard register automaton corresponds to $S = \{id\}$.

The following two propositions guarantee that a single orbit nominal set and an equivariant function between them have finite representations.

Proposition 1 ([3, Proposition 9.15]).

1. $\llbracket C, S \rrbracket$ is a single orbit nominal set.
2. Every single orbit nominal set is isomorphic to some $\llbracket C, S \rrbracket$.

$\llbracket C, S \rrbracket$ is called a *support representation* of a single orbit nominal set. The following proposition can be shown by [3, Proposition 9.16].

Proposition 2. *Let $X = \llbracket C, S \rrbracket$ and $Y = \llbracket D, T \rrbracket$ be single orbit nominal sets. For every equivariant function $f : X \rightarrow Y$, there is an injection u from D to C satisfying $uS \subseteq Tu$ and $f([\pi|_C]_S) = [u]_T \cdot \pi$, for all $\pi \in G$, where $uS = \{us \mid s \in S\}$ and $Tu = \{tu \mid t \in T\}$. Conversely, for every injection $u : D \rightarrow C$ satisfying $uS \subseteq Tu$, $f([\pi|_C]_S) = [u]_T \cdot \pi$ is an equivariant function from X to Y .*

A proof of this proposition is given in the Appendix.

By Proposition 2, we can obtain a necessary and sufficient condition for two single orbit nominal sets to be isomorphic in terms of a bijection between supports.

Lemma 1. *Single orbit nominal sets $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are isomorphic if and only if there exists a bijection $u : D \rightarrow C$ satisfying $uS = Tu$ that extends to a permutation from G .*

Proof. Assume that there exists a bijection $u : D \rightarrow C$ satisfying $uS = Tu$ that extends to a permutation from G . Then, we have $uS \subseteq Tu$ and $u^{-1}T \subseteq Su^{-1}$. By Proposition 2, we have two functions $f : \llbracket C, S \rrbracket \rightarrow \llbracket D, T \rrbracket$ and $g : \llbracket D, T \rrbracket \rightarrow \llbracket C, S \rrbracket$ such that

$$\begin{aligned} f([\pi|_C]_S) &= [u]_T \cdot \pi \\ g([\sigma|_D]_T) &= [u^{-1}]_S \cdot \sigma. \end{aligned}$$

We show that g is the inverse of f . From the assumption on u , there is some $\rho \in G$ satisfying $\rho|_D = u$ (and $\rho^{-1}|_C = u^{-1}$). Thus, $f([\pi|_C]_S) = [u]_T \cdot \pi = [\rho|_D]_T \cdot \pi = [(\rho\pi)|_D]_T$. Substituting this into the definition of g yields $g(f([\pi|_C]_S)) = [u^{-1}]_S \cdot (\rho\pi) = [\rho^{-1}|_C]_S \cdot (\rho\pi) = [(\rho^{-1}\rho\pi)|_C]_S = [\pi|_C]_S$. We have $g(f([\pi|_C]_S)) = [\pi|_C]_S$, and thus g is the inverse of f . Therefore, $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are isomorphic.

Conversely, assume that $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are isomorphic, i.e., there exists some equivariant bijection f from $\llbracket C, S \rrbracket$ to $\llbracket D, T \rrbracket$. By Proposition 2, f can be

written as $f([\pi|_C]_S) = [\sigma|_D]_T \cdot \pi$ for some $\sigma \in G$, such that there is an injection $u : D \rightarrow C$ satisfying $\sigma|_D = u$ and $uS \subseteq Tu$. Also by Proposition 2, f^{-1} can be written as $f^{-1}([\pi|_D]_T) = [\rho|_C]_S \cdot \pi$ for some $\rho \in G$, such that there is an injection $v : C \rightarrow D$ satisfying $\rho|_C = v$ and $vT \subseteq Sv$. From this, we can derive $f^{-1}([\sigma|_D]_T \cdot \pi) = f^{-1}([\pi|_D]_T) = [\rho|_C]_S \cdot \pi$. By $f([\pi|_C]_S) = [\sigma|_D]_T \cdot \pi$, we have $[\rho|_C]_S \cdot (\sigma\pi) = [\pi|_C]_S$, and hence $[\rho|_C]_S = [\pi|_C]_S \cdot (\sigma\pi)^{-1} = [(\sigma|_C)^{-1}]_S$. This means that $[v]_S = [u^{-1}]_S$. Thus, we have $vu \in S$. In the same way, we have $uv \in T$. By acting u on $vT \subseteq Sv$, we have $uvTu \subseteq uSvu$, and hence we have $Tu \subseteq uS$ by $vu \in S$ and $uv \in T$. By $uS \subseteq Tu$ and $Tu \subseteq uS$, $uS = Tu$. \square

Let X and Y be nominal sets. We define $Y \preceq X$ if and only if there is an **equivariant surjection** from a subset of X to Y . If $Y \preceq X$ and X and Y are not isomorphic, $Y \prec X$. We show that there is no infinite chain between two orbit finite nominal sets X and Y such that $Y \preceq X$. This property is used for proving the termination of the proposed learning algorithm. We start with X and Y being single orbits.

Lemma 2. *Let (\mathbb{D}, G) be a data symmetry that admits least support. Then, for any two single orbit nominal sets X and Y such that $Y \prec X$, the length of any sequence of single orbit nominal sets X_1, X_2, \dots satisfying*

$$Y \prec X_1 \prec X_2 \prec \dots \prec X$$

is finite.

Proof. By Proposition 1, it suffices to show the lemma for $X = \llbracket C, S \rrbracket$ and $Y = \llbracket D, T \rrbracket$. Assume $\llbracket D, T \rrbracket \prec \llbracket C, S \rrbracket$. By definition of \prec , there exists an equivariant surjective function from a subset of $\llbracket C, S \rrbracket$ to $\llbracket D, T \rrbracket$. The domain of this function is $\llbracket C, S \rrbracket$ because $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are single orbit sets. Thus, by Proposition 2, there exists an injection $u : D \rightarrow C$ satisfying $uS \subseteq Tu$ that extends to a permutation from G . Because u is an injection, $|D| \leq |C|$ holds where $|D|$ is the number of elements of D . If $|D| < |C|$, no injections from D to C are bijections, and hence $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are not isomorphic by Lemma 1. If $|D| = |C|$, then $uS \subsetneq Tu$ must hold because $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are not isomorphic. Thus, we have $S < u^{-1}Tu$. This means that S is a proper subgroup of $u^{-1}Tu$ and thus $|S| < |T|$. Therefore, because C and D are finite sets and S and T are finite groups, the length of any sequence of single orbit nominal sets X_1, X_2, \dots satisfying

$$Y \prec X_1 \prec X_2 \prec \dots \prec X$$

is finite. \square

Lemma 3. *Let (\mathbb{D}, G) be a data symmetry that admits least support. Then, for any two orbit finite nominal sets X and Y , the length of any sequence of orbit finite nominal sets X_1, X_2, \dots satisfying*

$$Y \prec X_1 \prec X_2 \prec \dots \prec X$$

is finite.

Proof. Any orbit finite nominal set is an union of a finite number of single orbit nominal sets. Hence, this lemma obviously holds by Lemma 2. \square

2.2 Data tree

Let (\mathbb{D}, G) be a data symmetry and A be an alphabet. We define an m -ary data tree (simply tree) over A as a function $t : Pos(t) \rightarrow A$ satisfying the following two conditions:

- $Pos(t) \subseteq \{1, \dots, m\}^*$ is a non-empty finite set that is prefix-closed, and
- every $p \in Pos(t)$ has a non-negative integer $arity(p) \leq m$ satisfying

$$pi \in Pos(t) \quad \text{for all } i \in \{1, \dots, arity(p)\},$$

where pi is the concatenation of p and i . The set of all m -ary data trees over A is written as $Tree_m(A)$.

We define subtree $t|_p$ of t at $p \in Pos(t)$ as

- $Pos(t|_p) = \{q \in \{1, \dots, m\}^* \mid pq \in Pos(t)\}$, and
- $t|_p(q) = t(pq)$ for all $q \in Pos(t|_p)$.

The set of all subtrees of t is written as $Subtree(t)$. To denote a tree, we will use term representation, which is recursively defined as follows. For $a \in A$ and terms trm_1, \dots, trm_k with $0 \leq k \leq m$, the term $a(trm_1, \dots, trm_k)$ represents the tree t such that $arity(\varepsilon) = k$ and

$$t(p) = \begin{cases} a & \text{if } p = \varepsilon, \\ t_i(q) & \text{if } p = iq \text{ for } 1 \leq i \leq k, \end{cases}$$

where t_i is the tree represented by trm_i for $1 \leq i \leq k$.

The group action on $Tree_m(A)$ is defined as $t \cdot \pi = (a \cdot \pi)(t_1 \cdot \pi, \dots, t_k \cdot \pi)$ for all $t \in Tree_m(A)$ and $\pi \in G$. $Tree_m(A)$ is a nominal set.

Let $x \notin A$ be a variable. A tree $t \in Tree_m(A \cup \{x\})$ is called a *context* of A if and only if there is exactly one $p \in Pos(t)$ satisfying $t(p) = x$ and $arity(p) = 0$. The set of all contexts of A is written as $Context_m(A)$. For $s \in Context_m(A)$ and $t \in (Tree_m(A) \cup Context_m(A))$, we define $s[t]$ as a tree with x in s replaced by t , i.e.,

$$s[t](p) = \begin{cases} s(p) & \text{if } p \in Pos(s) \text{ and } s(p) \neq x, \\ t(q) & \text{if } p = rq, s(r) = x \text{ and } q \in Pos(t). \end{cases}$$

For $S \subseteq Context_m(A)$ and $T \subseteq (Tree_m(A) \cup Context_m(A))$, we define $S[T] = \{s[t] \mid s \in S \text{ and } t \in T\}$. For all $\pi \in G$, we define $x \cdot \pi = x$.

Example 1. Let $c \in Context_m$ be a context of \mathbb{N} such that $Pos(c) = \{\varepsilon, 1\}$, $c(\varepsilon) = 2$, $c(1) = x$. Let $t \in Tree_2(\mathbb{N})$ be a 2-ary tree over \mathbb{N} such that $Pos(t) = \{\varepsilon, 1, 2\}$, $t(\varepsilon) = 3$, $t(1) = 1$, $t(2) = 5$. For c and t , $c[t]$ is the tree such that $Pos(c[t]) = \{\varepsilon, 1, 11, 12\}$, $c[t](\varepsilon) = 2$, $c[t](1) = 3$, $c[t](11) = 1$, $c[t](12) = 5$. Figure 1 illustrates c , t and $c[t]$. Term representations of c , t and $c[t]$ are $c = 2(x)$, $t = 3(1, 5)$ and $c[t] = 2(3(1, 5))$, respectively. The term representation of subtree $c[t]_{12}$ of $c[t]$ at $12 \in Pos(c[t])$ is 5.

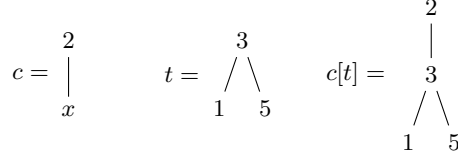


Fig. 1. Data trees and a context over \mathbb{N} .

3 Deterministic bottom-up nominal tree automata

Let (\mathbb{D}, G) be a data symmetry and A be an alphabet. A *deterministic bottom-up tree automaton* (G -DBTA) over $Tree_m(A)$ is a triple $\mathcal{A} = (Q, F, \delta)$, where

- Q is a G -set of states,
- $F \subseteq Q$ is an equivariant set of accept states, and
- $\delta = (\delta_0, \dots, \delta_m)$ is an $m + 1$ -tuple of equivariant transition functions, where

$$\begin{aligned} \delta_0 &: A \rightarrow Q, \\ \delta_k &: A \times Q^k \rightarrow Q \quad \text{for } 1 \leq k \leq m. \end{aligned}$$

We extend δ to the function on $Tree_m(A)$ by

$$\delta(a(t_1, \dots, t_k)) = \begin{cases} \delta_k(a, \delta(t_1), \dots, \delta(t_k)) & \text{if } k > 0, \\ \delta_0(a) & \text{if } k = 0. \end{cases}$$

A tree t is accepted by \mathcal{A} if and only if $\delta(t) \in F$. We define $L(\mathcal{A}) = \{t \in Tree_m(A) \mid \delta(t) \in F\}$. We call $L \subseteq Tree_m(A)$ a *recognizable tree language* when there exists a G -DBTA \mathcal{A} satisfying $L = L(\mathcal{A})$. A G -DBTA $\mathcal{A} = (Q, F, \delta)$ is *reachable* if for all $q \in Q$, there exists some $t \in Tree_m(A)$ such that $\delta(t) = q$. If A and Q are orbit finite nominal sets, then \mathcal{A} is called a *deterministic bottom-up nominal tree automaton* (G -DBNTA). We call $L \subseteq Tree_m(A)$ a *recognizable nominal tree language* when there exists a G -DBNTA \mathcal{A} satisfying $L = L(\mathcal{A})$.

Let $X \subseteq Tree_m(A)$ be a subset of trees. For a function $T : X \rightarrow \{0, 1\}$, a G -DBTA \mathcal{A} is *consistent with T* if for all $t \in X$, $t \in L(\mathcal{A}) \Leftrightarrow T(t) = 1$.

Example 2. Let the set \mathbb{N} of natural numbers be an alphabet and $(\mathbb{N}, \text{Sym}(\mathbb{N}))$ be a data symmetry. Let $\mathcal{A} = (Q, F, \delta)$ be a G -DBTA over $Tree_2(\mathbb{N})$, where $Q = \mathbb{N} \cup \{\text{accept}, \text{reject}\}$, $F = \{\text{accept}\}$ and $\delta = (\delta_0, \delta_1, \delta_2)$ such that $\delta_0(d) = d$, $\delta_1(d, q) = \text{reject}$ and $\delta_2(d, q_1, q_2) = \text{accept}$ if $q_1 = q_2 = d$, *reject* otherwise. For all $\pi \in \text{Sym}(\mathbb{N})$, we define $\text{accept} \cdot \pi = \text{accept}$ and $\text{reject} \cdot \pi = \text{reject}$. It is easy to see that all components of \mathcal{A} are equivariant. The tree language recognized by \mathcal{A} is $L(\mathcal{A}) = \{d(d, d) \mid d \in \mathbb{N}\}$.

4 Myhill-Nerode theorem

Let (\mathbb{D}, G) be a data symmetry that admits least support and A be a nominal alphabet. For $L \subseteq \text{Tree}_m(A)$, we define the binary relation \approx_L over $\text{Tree}_m(A)$ as follows: $u \approx_L v$ if and only if

$$C[u] \in L \text{ iff } C[v] \in L \text{ for all } C \in \text{Context}_m(A).$$

It is easy to check that \approx_L is an congruence relation on $\text{Tree}_m(A)$, i.e., \approx_L is an equivalent relation that satisfies $a(u_1, \dots, u_k) \approx_L a(v_1, \dots, v_k)$ for all $a \in A$ and $u_1, \dots, u_k, v_1, \dots, v_k \in \text{Tree}_m(A)$ with $u_i \approx_L v_i$ for $0 \leq i \leq k$. We write the equivalent class of $t \in \text{Tree}_m(A)$ as $[t]$.

Lemma 4. *If $L \subseteq \text{Tree}_m(A)$ is equivariant, then \approx_L is also equivariant.*

Proof. We show that $t \cdot \pi \approx_L t' \cdot \pi$ for all $\pi \in G$ and $t, t' \in \text{Tree}_m(A)$ such that $t \approx_L t'$. By the definition of \approx_L , $t \cdot \pi \approx_L t' \cdot \pi$ is equivalent to $C[t \cdot \pi] \in L$ iff $C[t' \cdot \pi] \in L$ for all $C \in \text{Context}_m(A)$. By the equivariance of L , this is equivalent to $C[t \cdot \pi] \cdot \pi^{-1} \in L$ iff $C[t' \cdot \pi] \cdot \pi^{-1} \in L$. By the definition of the group action on $\text{Tree}_m(A)$, this is equivalent to $(C \cdot \pi^{-1})[t] \in L$ iff $(C \cdot \pi^{-1})[t'] \in L$. We can prove this by $t \approx_L t'$. \square

Lemma 5 ([3, Lemma 3.5]). *Let X be a G -set and $R \subseteq X \times X$ be an equivalence relation that is equivariant. Then the quotient X/R is a G -set, under the action $[x]_R \cdot \pi = [x \cdot \pi]_R$ of G , and the abstraction mapping $x \mapsto [x]_R : X \rightarrow X/R$ is an equivariant function.*

For $L \subseteq \text{Tree}_m(A)$, we define the *syntactic tree automaton* $\mathcal{A}_L = (Q_L, F_L, \delta_L)$ as

- $Q_L = \text{Tree}_m(A)/\approx_L$,
- $F_L = \{[t] \mid t \in L\}$ and
- $\delta_L = (\delta_{L,0}, \dots, \delta_{L,m})$ where $\delta_{L,0} : A \rightarrow Q_L$ and $\delta_{L,k} : A \times Q_L^k \rightarrow Q_L$ for $1 \leq k \leq m$ are defined as

$$\begin{aligned} \delta_{L,0}(a) &= [a], \\ \delta_{L,k}(a, [u_1], \dots, [u_k]) &= [a(u_1, \dots, u_k)]. \end{aligned}$$

Because \approx_L is a congruence relation, δ_L is well-defined.

Lemma 6. *If $L \subseteq \text{Tree}_m(A)$ is equivariant, then the syntactic tree automaton $\mathcal{A}_L = (Q_L, F_L, \delta_L)$ is a reachable G -DBTA.*

Proof. Because L is equivariant, \approx_L is also equivariant by Lemma 4. Thus, by Lemma 5, $Q_L = \text{Tree}_m(A)/\approx_L$ is a G -set. By the equivariance of L ,

$$[t] \in F_L \Leftrightarrow t \in L \Leftrightarrow t \cdot \pi \in L \Leftrightarrow [t \cdot \pi] \in F_L \Leftrightarrow [t] \cdot \pi \in F_L.$$

Thus, F_L is equivariant. By $\delta_L(a, [u_1], \dots, [u_k]) \cdot \pi = [a(u_1, \dots, u_k)] \cdot \pi = [(a \cdot \pi)(u_1 \cdot \pi, \dots, u_k \cdot \pi)] = \delta_L(a \cdot \pi, [u_1 \cdot \pi], \dots, [u_k \cdot \pi]) = \delta_L(a \cdot \pi, [u_1] \cdot \pi, \dots, [u_k] \cdot \pi)$, δ_L is equivariant. Thus, \mathcal{A}_L is a G -DBTA. \mathcal{A}_L is apparently reachable. \square

Let $\mathcal{A} = (Q, F, \delta)$ and $\mathcal{A}' = (Q', F', \delta')$ be G -DBTAs. An equivariant function $\varphi : P \rightarrow Q'$ for some $P \subseteq Q$ satisfying the following two conditions is called a *partial homomorphism* from \mathcal{A} to \mathcal{A}' :

- $q \in F$ iff $\varphi(q) \in F'$ for all $q \in P$, and
- $\varphi(\delta(a, q_1, \dots, q_k)) = \delta'(a, \varphi(q_1), \dots, \varphi(q_k))$ for all $q_1, \dots, q_k \in P$ ($0 \leq k \leq m$) and $a \in A$.

When there exists a surjective partial homomorphism from a subset of Q to Q' , we write $\mathcal{A}' \sqsubseteq \mathcal{A}$. If $P = Q$, then φ is called a *homomorphism* from \mathcal{A} to \mathcal{A}' . It is easy to see that $L(\mathcal{A}) = L(\mathcal{A}')$ if there is a homomorphism φ from \mathcal{A} to \mathcal{A}' . When φ is surjective, \mathcal{A}' is called an image of \mathcal{A} . If \mathcal{A}' is an image of \mathcal{A} and the state set of \mathcal{A} is orbit finite, the state set of \mathcal{A}' is also orbit finite.

Lemma 7. *Let L be a recognizable tree language. The syntactic automaton A_L is an image of any reachable G -DBTA recognizing L .*

Proof. Let $\mathcal{A} = (Q, F, \delta)$ be a reachable G -DBTA recognizing L . We define $\varphi : Q \rightarrow \text{Tree}_m(A)/\approx_L$ as $\varphi(\delta(t)) = [t]$. The definition of φ is well-defined because \mathcal{A} is reachable and $\delta(u) = \delta(v)$ implies $[u] = [v]$. It is easy to check that φ is surjective. By the equivariance of δ ,

$$\varphi(\delta(t) \cdot \pi) = \varphi(\delta(t \cdot \pi)) = [t \cdot \pi] = [t] \cdot \pi = \varphi(\delta(t)) \cdot \pi.$$

Thus, φ is equivariant. We have

$$\begin{aligned} \varphi(\delta(a(u_1, \dots, u_k))) &= [a(u_1, \dots, u_k)] \\ &= \delta_L(a, [u_1], \dots, [u_k]) = \delta(a, \varphi(\delta(u_1)), \dots, \varphi(\delta(u_k))). \end{aligned}$$

We also have $\delta(t) \in F \Leftrightarrow t \in L \Leftrightarrow [t] \in F_L$. Therefore, φ is a homomorphism, and hence A_L is an image of \mathcal{A} . \square

Let $\mathcal{A} = (Q, F, \delta)$ be a reachable G -DBTA over $\text{Tree}_m(A)$. The equivariant function $t \mapsto \delta(t)$ from $\text{Tree}_m(A)$ to Q is surjective because \mathcal{A} is reachable. If $C \subseteq \mathbb{D}$ supports t , then C also supports $\delta(t)$ because $t \cdot \pi = t$ implies $\delta(t) \cdot \pi = \delta(t \cdot \pi) = \delta(t)$ for all $\pi \in G$. Thus, Q is nominal because $\text{Tree}_m(A)$ is nominal.

Theorem 1. *Let $L \subseteq \text{Tree}_m(A)$ be an equivariant set. The following two conditions are equivalent:*

- (1) $\text{Tree}_m(A)/\approx_L$ is orbit finite.
- (2) L is recognized by a G -DBNTA.

Proof. (1) \Rightarrow (2) can be easily proved by Lemma 6. Without loss of generality, assume that a given G -DBNTA is reachable. (2) \Rightarrow (1) can be proved by Lemma 7. \square

5 Observation table

In this and the next sections, we extend the L^* -style algorithm in [17] to DBNTA. For the extension from words to trees, we extend some notions given in [21], where another L^* -style algorithm is proposed to learn the set of derivation trees of an unknown context-free grammar without data values.

From now on, we assume a data symmetry (\mathbb{D}, G) that admits least support and a nominal alphabet A . Let $B \subseteq \text{Tree}_m(A)$ and $C \subseteq \text{Context}_m(A)$. We say that B is *subtree-closed* if and only if for every $b \in B$, $\text{Subtree}(b) \subseteq B$ holds. We also say that C is *x -prefix-closed* on B if and only if every $c \in C \setminus \{x\}$ has some $c' \in C$ satisfying $c = c'[a(b_1, \dots, b_{i-1}, x, b_i, \dots, b_{k-1})]$ where $b_1, \dots, b_{k-1} \in B$ and $a \in A$.

Definition 1. Let U be an unknown recognizable nominal tree language. An observation table is a triple $(\mathcal{S}, \mathcal{E}, \mathcal{T})$, where

- $\mathcal{S} \subseteq \text{Tree}_m(A)$ is an equivariant orbit finite set that is subtree-closed and satisfies $A \subseteq \mathcal{S}$.
- $\text{Next}(\mathcal{S}) = \{a(t_1, \dots, t_k) \notin \mathcal{S} \mid a \in A, t_1, \dots, t_k \in \mathcal{S}, 1 \leq k \leq m\}$,
- $\mathcal{E} \subseteq \text{Context}_m(A)$ is an equivariant orbit finite set that is x -prefix-closed on \mathcal{S} , and
- $\mathcal{T} : \mathcal{E}[\mathcal{S} \cup \text{Next}(\mathcal{S})] \rightarrow \{0, 1\}$ is an equivariant function, where $\mathcal{T}(e[s]) = 1$ iff $e[s] \in U$ for all $e \in \mathcal{E}$ and $s \in \mathcal{S} \cup \text{Next}(\mathcal{S})$. \square

We define the function $\text{row}_{(\mathcal{S}, \mathcal{E}, \mathcal{T})} : \mathcal{S} \cup \text{Next}(\mathcal{S}) \rightarrow 2^{\mathcal{E}}$ as $\text{row}_{(\mathcal{S}, \mathcal{E}, \mathcal{T})}(s) = \{e \in \mathcal{E} \mid \mathcal{T}(e[s]) = 1\}$. We abbreviate $\text{row}_{(\mathcal{S}, \mathcal{E}, \mathcal{T})}$ as row if $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is clear from the context. It is easy to see that row is equivariant. For $X \subseteq \mathcal{S} \cup \text{Next}(\mathcal{S})$, we define $\text{row}(X) = \{\text{row}(s) \mid s \in X\}$.

An observation table can be expressed by the table with rows labeled with the elements of $\mathcal{S} \cup \text{Next}(\mathcal{S})$ and columns labeled with the elements of \mathcal{E} as shown in Fig. 2.

		\mathcal{E} e
\mathcal{S}	s	\vdots $\dots \quad \mathcal{T}(e[s])$
$\text{Next}(\mathcal{S})$		

Fig. 2. Observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$

An observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is *closed* if and only if for all $t \in \text{Next}(\mathcal{S})$, there exists some $s \in \mathcal{S}$ satisfying $\text{row}(t) = \text{row}(s)$. An observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is *consistent* if and only if for every $s_1, s_2 \in \mathcal{S}$, $\text{row}(s_1) = \text{row}(s_2)$ implies

$$\text{row}(a(u_1, \dots, u_{i-1}, s_1, u_i, \dots, u_{k-1})) = \text{row}(a(u_1, \dots, u_{i-1}, s_2, u_i, \dots, u_{k-1}))$$

for all $a \in A, u_1, \dots, u_{k-1} \in \mathcal{S}$ and $1 \leq i \leq k$.

Let $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ be a closed and consistent observation table. We define the G -DBNTA $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T}) = (\tilde{Q}, \tilde{F}, \tilde{\delta})$ derived from $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ as follows:

- $\tilde{Q} = \text{row}(\mathcal{S}) = \{\text{row}(s) \mid s \in \mathcal{S}\}$,
- $\tilde{F} = \{\text{row}(s) \mid s \in \mathcal{S}, \mathcal{T}(s) = 1\}$,
- $\tilde{\delta}_k(a, \text{row}(s_1), \dots, \text{row}(s_k)) = \text{row}(a(s_1, \dots, s_k))$ for $s_1, \dots, s_k \in \mathcal{S}$.

It is easy to see that $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is well-defined: Let $s_1, s_2 \in \mathcal{S}$ be trees satisfying $\text{row}(s_1) = \text{row}(s_2)$. Because \mathcal{E} is x -prefix-closed, $x \in \mathcal{E}$ holds. Thus, $\mathcal{T}(s_1) = \mathcal{T}(x[s_1])$ and $\mathcal{T}(s_2) = \mathcal{T}(x[s_2])$ are defined, and $\mathcal{T}(s_1) = \mathcal{T}(s_2)$, and so \tilde{F} is well-defined. Because $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is consistent,

$$\text{row}(a(u_1, \dots, u_{i-1}, s_1, u_i, \dots, u_{k-1})) = \text{row}(a(u_1, \dots, u_{i-1}, s_2, u_i, \dots, u_{k-1}))$$

for all $a \in A, u_1, \dots, u_{k-1} \in \mathcal{S}$ and $1 \leq i \leq k$. Moreover, because $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is closed, there is $s \in \mathcal{S}$ satisfying $\text{row}(s) = \text{row}(a(u_1, \dots, u_{i-1}, s_1, u_i, \dots, u_{k-1}))$. Therefore, $\tilde{\delta}$ is well-defined. Because \mathcal{S} is an orbit finite nominal set and row is an equivariant function, $\tilde{Q}(= \text{row}(\mathcal{S}))$ is also an orbit finite nominal set.

Lemma 8. *Let $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ be a closed and consistent observation table. Then, $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T}) = (\tilde{Q}, \tilde{F}, \tilde{\delta})$ is consistent with \mathcal{T} .*

The proof is similar to the proof of Lemma 4.2 in [21].

Theorem 2. *For a (not necessarily closed and consistent) observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ and a G -DBNTA $\mathcal{A} = (Q, F, \delta)$ consistent with \mathcal{T} , $\text{row}(\mathcal{S}) \preceq Q$ holds.*

Proof. We show that the function $\delta(s) \mapsto \text{row}(s)$ is an equivariant surjection from $\{\delta(s) \mid s \in \mathcal{S}\} \subseteq Q$ to $\text{row}(\mathcal{S})$. This function is well-defined since

$$\begin{aligned} \delta(s_1) = \delta(s_2) &\Rightarrow \forall e \in \mathcal{E}. \delta(e[s_1]) = \delta(e[s_2]) \\ &\Rightarrow \forall e \in \mathcal{E}. e[s_1] \in L(\mathcal{A}) \quad \text{iff} \quad e[s_2] \in L(\mathcal{A}) \\ &\Leftrightarrow \forall e \in \mathcal{E}. \mathcal{T}(e[s_1]) = \mathcal{T}(e[s_2]) \\ &\Leftrightarrow \text{row}(s_1) = \text{row}(s_2). \end{aligned}$$

This function is also equivariant because $\delta(s) \cdot \pi = \delta(s \cdot \pi) \mapsto \text{row}(s \cdot \pi) = \text{row}(s) \cdot \pi$. Surjectivity is clear. Therefore, because there is an equivariant function from a subset of Q to $\text{row}(\mathcal{S})$, $\text{row}(\mathcal{S}) \preceq Q$ holds. \square

If an observation table is closed and consistent, Theorem 2 can be lifted from a relation on states (\preceq) to a relation on automata (\sqsubseteq) as stated in the next lemma.

Lemma 9. *Let $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ be a closed and consistent observation table. For every G -DBNTA \mathcal{A} that is consistent with \mathcal{T} , $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T}) \sqsubseteq \mathcal{A}$ holds.*

Proof. Let $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T}) = (\tilde{Q}, \tilde{F}, \tilde{\delta})$ and $A = (Q, F, \delta)$. By the proof of Theorem 2, the function $\varphi : \delta(s) \mapsto \text{row}(s)$ from $\{\delta(s) \mid s \in \mathcal{S}\} \subseteq Q$ to $\tilde{Q} (= \text{row}(\mathcal{S}))$ is equivariant and surjective. By $\delta(s) \in F \Leftrightarrow \mathcal{T}(s) = 1 \Leftrightarrow \text{row}(s) \in \tilde{F} \Leftrightarrow \varphi(\delta(s)) \in \tilde{F}$ and $\varphi(\delta(a, \delta(s_1), \dots, \delta(s_k))) = \varphi(\delta(a(s_1, \dots, s_k))) = \text{row}(a(s_1, \dots, s_k)) = \tilde{\delta}(a, \text{row}(s_1), \dots, \text{row}(s_k)) = \tilde{\delta}(a, \varphi(\delta(s_1)), \dots, \varphi(\delta(s_k)))$, φ is a partial homomorphism. \square

By Lemmas 8 and 9, we have the following theorem.

Theorem 3. *Let $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ be a closed and consistent observation table. $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T})$ is consistent with \mathcal{T} , and for every G -DBNTA \mathcal{A} that is consistent with \mathcal{T} , $\mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T}) \sqsubseteq \mathcal{A}$ holds.*

6 Learning algorithm

We show the proposed learning algorithm (Algorithm 1) in the following page. We will give a part of a run of Algorithm 1 on an example in Section 7. In the Algorithm 1, we assume that the teacher answering queries is given as an oracle. In an application to the compositional verification, for example, the teacher is implemented as a model checker (see [8]).

Because \mathcal{S} and \mathcal{E} of an observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ can be infinite sets, we have to show that $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ can be expressed by finite means and each step of Algorithm 1 runs in finite steps. We first show that $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ has a finite description. Because \mathcal{S} and \mathcal{E} are orbit finite nominal sets, by Proposition 1, we can express \mathcal{S} and \mathcal{E} by support representations. By Proposition 2, we can express \mathcal{T} by finite means because \mathcal{T} consists of a finite number of equivariant functions whose domains and ranges are both single orbit nominal sets. In Algorithm 1, each orbit O is represented by any one element $s \in O$. Let us call s a representative of O .

Next, we show that each step of Algorithm 1 runs in finite steps. To check the closedness in line 10 of Algorithm 1, it suffices to check whether for each orbit O of $\text{Next}(\mathcal{S})$ and a representative s' of O , there is $s \in \mathcal{S}$ such that $\text{row}(s) = \text{row}(s')$. Finding $s \in \mathcal{S}$ satisfying $\text{row}(s) = \text{row}(s')$ is equivalent to finding $\pi \in G$ such that $\text{row}(s') = \text{row}(t \cdot \pi) (= \text{row}(t) \cdot \pi)$ for some representative $t \in \mathcal{S}$. Let $C, D \subseteq \mathbb{D}$ be the least supports of $\text{row}(s')$ and $\text{row}(t)$, respectively. The least support of $\text{row}(t) \cdot \pi$ is $D \cdot \pi$. Thus, if $\text{row}(s') = \text{row}(t \cdot \pi)$, then $C = D \cdot \pi$ must hold. Moreover, because D is the (least) support of $\text{row}(t)$, if $\pi_1|_D = \pi_2|_D$ then $\text{row}(t) \cdot \pi_1 = \text{row}(t) \cdot \pi_2$. Thus, we only have to check a finite number of π satisfying $C = D \cdot \pi$. To check the consistency in line 5 of Algorithm 1, it suffices to check the emptiness of

$$\{(s_1, s_2, a, e) \in \mathcal{S} \times \mathcal{S} \times A \times \mathcal{E} \mid \text{row}(s_1) = \text{row}(s_2) \text{ and for } \exists u_1, \dots, u_{k-1} \in \mathcal{S}, \\ \mathcal{T}(e[a(u_1, \dots, u_{i-1}, s_1, u_i, \dots, u_{k-1})]) \neq \mathcal{T}(e[a(u_1, \dots, u_{i-1}, s_2, u_i, \dots, u_{k-1})])\}.$$

Algorithm 1 Angluin-style algorithm for $G\text{-DBNTA}$

```

1:  $\mathcal{S} := A, \mathcal{E} := \{x\}$ ;
2: Construct the initial observation table  $(\mathcal{S}, \mathcal{E}, \mathcal{T})$  using membership queries;
3: repeat
4:   while  $(\mathcal{S}, \mathcal{E}, \mathcal{T})$  is not closed or not consistent do
5:     if  $(\mathcal{S}, \mathcal{E}, \mathcal{T})$  is not consistent then
6:       Find  $s_1, s_2, u_1, \dots, u_{k-1} \in \mathcal{S}, e \in \mathcal{E}, a \in A, i \in \mathbb{N}$  such that
          $row(s_1) = row(s_2)$  and
          $T(e[a(u_1, \dots, u_{i-1}, s_1, u_i, \dots, u_k)]) \neq T(e[a(u_1, \dots, u_{i-1}, s_2, u_i, \dots, u_k)])$ ;
7:       Add  $Orbit(e[a(u_1, \dots, u_{i-1}, x, u_i, \dots, u_k)])$  to  $\mathcal{E}$ ;
8:       Extend  $\mathcal{T}$  to  $\mathcal{E}[(\mathcal{S} \cup Next(\mathcal{S}))]$  using membership queries;
9:     end if
10:    if  $(\mathcal{S}, \mathcal{E}, \mathcal{T})$  is not closed then
11:      Find  $s' \in Next(\mathcal{S})$  such that  $row(s') \neq row(s)$  for all  $s \in \mathcal{S}$ ;
12:      Add  $Orbit(s')$  to  $\mathcal{S}$ ;
13:      Extend  $\mathcal{T}$  to  $\mathcal{E}[\mathcal{S} \cup Next(\mathcal{S})]$  using membership queries;
14:    end if
15:  end while
16:  Let  $\mathcal{A} = \mathcal{A}(\mathcal{S}, \mathcal{E}, \mathcal{T})$ ;
17:  Construct the conjecture  $\mathcal{A}$ ;
18:  if the Teacher replies no with a counter-example  $t$  then
19:    Add  $Orbit(Subtree(t))$  to  $\mathcal{S}$ ;
20:    Extend  $\mathcal{T}$  to  $\mathcal{E}[\mathcal{S} \cup Next(\mathcal{S})]$  using membership queries;
21:  end if
22: until the Teacher replies yes to the conjecture  $\mathcal{A}$ ;
23: return  $\mathcal{A}$ ;

```

$\mathcal{S} \times \mathcal{S} \times A \times \mathcal{E}$ is an orbit finite nominal set, and the above set is a union of some orbits of $\mathcal{S} \times \mathcal{S} \times A \times \mathcal{E}$. Thus, we can check the emptiness of the set, and if not, we can obtain representatives of the set.

Correctness Because Algorithm 1 uses an equivalence query, if it terminates, then it outputs the correct $G\text{-DBNTA}$.

To prove the termination of Algorithm 1, we show the following two lemmas that guarantee that $row(\mathcal{S})$ strictly increases with respect to \prec each time an observation table is extended.

Lemma 10. *If $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ and $(\mathcal{S}', \mathcal{E}, \mathcal{T}')$ are observation tables such that $\mathcal{S} \subsetneq \mathcal{S}'$ and $\mathcal{T}'(e[s]) = \mathcal{T}(e[s])$ for all $e \in \mathcal{E}$ and $s \in \mathcal{S}$, then $row_{(\mathcal{S}, \mathcal{E}, \mathcal{T})}(\mathcal{S}) \prec row_{(\mathcal{S}', \mathcal{E}, \mathcal{T}')}(\mathcal{S}')$.*

Proof. The lemma obviously holds because if $\mathcal{S} \subsetneq \mathcal{S}'$, the number of orbits of \mathcal{S}' is larger than that of \mathcal{S} . \square

Lemma 11. *If $(\mathcal{S}, \mathcal{E}, \mathcal{T})$ and $(\mathcal{S}, \mathcal{E}', \mathcal{T}')$ are observation tables such that $\mathcal{E} \subsetneq \mathcal{E}'$ and $\mathcal{T}'(e[s]) = \mathcal{T}(e[s])$ for all $e \in \mathcal{E}$ and $s \in \mathcal{S}$, then $row_{(\mathcal{S}, \mathcal{E}, \mathcal{T})}(\mathcal{S}) \prec row_{(\mathcal{S}, \mathcal{E}', \mathcal{T}')}(\mathcal{S})$.*

A proof of this lemma is given in the Appendix.

Termination and minimality Let U be an unknown recognizable nominal tree language and $\mathcal{A}_U = (Q_U, F_U, \delta_U)$ be the syntactic tree automaton constructed from U . \mathcal{A}_U is the minimum G -DBNTA recognizing U in the sense of Lemma 7. Let $(\mathcal{S}_0, \mathcal{E}_0, \mathcal{T}_0), (\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1), (\mathcal{S}_2, \mathcal{E}_2, \mathcal{T}_2), \dots$ be observation tables constructed by Algorithm 1 where $(\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i)$ extends to $(\mathcal{S}_{i+1}, \mathcal{E}_{i+1}, \mathcal{T}_{i+1})$ for $i \geq 0$. Note that \mathcal{A}_U is consistent with every \mathcal{T}_i for $i \geq 0$. By Lemmas 10 and 11, $\text{row}(\mathcal{S}_0) \prec \text{row}(\mathcal{S}_1) \prec \text{row}(\mathcal{S}_2) \prec \dots$. By Theorem 2, $\text{row}(\mathcal{S}_i) \preceq Q_U$ for $i \geq 0$. By Lemma 3, there is a non-negative integer n such that $\text{row}(\mathcal{S}_0) \prec \text{row}(\mathcal{S}_1) \prec \dots \prec \text{row}(\mathcal{S}_n) = Q_U$. Thus, Algorithm 1 terminates in finite steps. By Lemma 9, $\mathcal{A}(\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i) \subseteq \mathcal{A}_U$ holds for every $i \geq 0$ such that $(\mathcal{S}_i, \mathcal{E}_i, \mathcal{T}_i)$ is closed and consistent, and hence, Algorithm 1 outputs the minimum G -DBNTA recognizing U when it terminates.

Running time analysis When Algorithm 1 extends an observation table $(\mathcal{S}, \mathcal{E}, \mathcal{T})$, the number of orbits of $\text{row}(\mathcal{S})$ increases or some orbits of $\text{row}(\mathcal{S})$ extend. Extending an orbit $\llbracket C, S \rrbracket$ of $\text{row}(\mathcal{S})$ to $\llbracket D, T \rrbracket$ implies $|C| \leq |D|$. If $|C| = |D|$, then $T \leq uSu^{-1}$ must hold for some injection $u : D \rightarrow C$. By the standard theorem of finite groups, $|uSu^{-1}| (= |S|)$ can be divided by $|T|$. Therefore, we have the following theorem:

Theorem 4. *Let U be an unknown recognizable nominal tree language, $Q = \llbracket C_1, S_1 \rrbracket \cup \dots \cup \llbracket C_n, S_n \rrbracket$ be the set of states of the minimum G -DBNTA recognizing U , n be the number of orbits of Q and $m = \max\{|C_1|, \dots, |C_n|\}$ be the largest cardinality of least supports of orbits of Q . Let p_1, \dots, p_k be prime numbers and j_1, \dots, j_k be positive integers such that $m! = p_1^{j_1} \cdot p_2^{j_2} \cdot \dots \cdot p_k^{j_k}$. Observation tables are extended at most $O(nm(j_1 + \dots + j_k))$ times.*

7 Example

Let $(\mathbb{N}, \text{Sym}(\mathbb{N}))$ be the equality symmetry and $A = \mathbb{N}$ be an alphabet. Note that $(\mathbb{N}, \text{Sym}(\mathbb{N}))$ admits least support and A is a single orbit nominal set. Let $U = \text{Orbit}(1) \cup \text{Orbit}(1(1)) \cup \text{Orbit}(1(1(1))) \subseteq \text{Tree}_2(A)$. We now show a part of a run of Algorithm 1 for U .

First, the elements of the initial observation table $(\mathcal{S}_0, \mathcal{E}_0, \mathcal{T}_0)$ shown in Table 1 are $\mathcal{S}_0 = \text{Orbit}(1) (= \mathbb{N})$, $\mathcal{E}_0 = \{x\}$, $\text{Next}(\mathcal{S}_0) = \text{Orbit}(1(1)) \cup \text{Orbit}(2(1)) \cup \text{Orbit}(1(1, 1)) \cup \text{Orbit}(1(2, 1)) \cup \text{Orbit}(1(1, 2)) \cup \text{Orbit}(2(1, 1))$, $\mathcal{T}_0(a) = \mathcal{T}_0(a(a)) = 1$ and $\mathcal{T}_0(a(b)) = \mathcal{T}_0(a(a, a)) = \mathcal{T}_0(a(b, a)) = \mathcal{T}_0(a(a, b)) = \mathcal{T}_0(a(b, b)) = 0$ for all $a, b \in A$ such that $a \neq b$. This observation table $(\mathcal{S}_0, \mathcal{E}_0, \mathcal{T}_0)$ is consistent but not closed because there is no $s \in \mathcal{S}_0$ such that $\text{row}(s) = \text{row}(2(1))$. Thus, Algorithm 1 adds $\text{Orbit}(2(1))$ to \mathcal{S}_0 and extends \mathcal{T}_0 using membership queries. We have the observation table $(\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1)$ shown in Table 2 where

$$\begin{aligned} \mathcal{S}_1 &= \text{Orbit}(1) \cup \text{Orbit}(2(1)), \quad \mathcal{E}_1 = \{x\}, \\ \text{Next}(\mathcal{S}_1) &= \{a(t) \notin \mathcal{S}_1 \mid a \in A, t \in \mathcal{S}_1\} \cup \{a(t_1, t_2) \notin \mathcal{S}_1 \mid a \in A, t_1, t_2 \in \mathcal{S}_1\}. \end{aligned}$$

$(\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1)$ is closed and consistent, and Algorithm 1 asks an equivalence query with G -DBNTA $\mathcal{A}(\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1)$. $\mathcal{A}(\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1)$ does not recognize U because $1(1(1)) \notin L(\mathcal{A}(\mathcal{S}_1, \mathcal{E}_1, \mathcal{T}_1))$. Thus, Algorithm 1 adds $Orbit(1(1(1)))$ to \mathcal{S}_1 if $1(1(1))$ is returned as a counterexample and extends \mathcal{T}_1 using membership queries. We have the observation table $(\mathcal{S}_2, \mathcal{E}_2, \mathcal{T}_2)$ shown in Table 3. $(\mathcal{S}_2, \mathcal{E}_2, \mathcal{T}_2)$ is closed but not consistent because despite $row(1) = row(1(1(1)))$, $row(1(1)) \neq row(1(1(1(1))))$. Thus, Algorithm 1 adds $Orbit(1(x))$ to \mathcal{E}_2 and extends \mathcal{T}_2 using membership queries, resulting in the observation table $(\mathcal{S}_3, \mathcal{E}_3, \mathcal{T}_3)$ shown in Table 4. Continuing these extensions, Algorithm 1 finally obtains an observation table $(\mathcal{S}_n, \mathcal{E}_n, \mathcal{T}_n)$ such that $\mathcal{A}(\mathcal{S}_n, \mathcal{E}_n, \mathcal{T}_n)$ recognizes U .

Table 1.

	x
a	1
$a(a)$	1
$a(b)$	0
$a(a, a)$	0
$a(a, b)$	0
$a(b, a)$	0
$a(b, b)$	0

Table 2.

	x
a	1
$a(b)$	0
$a(a)$	1
$a(a, a)$	0
$a(a, b)$	0
$a(b, a)$	0
$a(b, b)$	0
<i>others</i>	0

Table 3.

	x
a	1
$a(b)$	0
$a(a(a))$	1
$a(a)$	1
$a(a, a)$	0
$a(a, b)$	0
$a(b, a)$	0
$a(b, b)$	0
<i>others</i>	0

Table 4.

	x	$c(x)$
a	1	1 ($a = c$)
	0	0 ($a \neq c$)
$a(b)$	0	0
$a(a(a))$	1	0
$a(a)$	1	1 ($a = c$)
	0	0 ($a \neq c$)
$a(a, a)$	0	0
$a(a, b)$	0	0
$a(b, a)$	0	0
$a(b, b)$	0	0
<i>others</i>	0	0

for all $a, b, c \in A$ satisfying $a \neq b$.

8 Conclusion

In this paper, we defined deterministic bottom-up nominal tree automata (DBNTA), which operate on trees whose nodes are labelled with elements of an orbit finite nominal set. We then proved a Myhill-Nerode theorem for the class of languages recognized by DBNTA and proposed an active learning algorithm for DBNTA based on the theorem. The algorithm can deal with any data symmetry that admits least support, not restricted to the equality symmetry and/or the total order symmetry.

Implementation and possible applications of the proposed learning algorithm are left as future work. For implementation, a concrete data structure for support representations of orbit finite sets in an observation table should be determined. Moreover, we are considering an application of the proposed algorithm to a compositional verification of a program that manipulates XML documents.

References

1. D. Angluin, Learning regular sets from queries and counterexamples, *Information and Computation* 75, 87–106, 1987.
2. M. Bojańczyk, C. David, A. Muscholl, T. Schwentick and L. Segoufin, Two-variable logic on data words, *ACM Trans. Computational Logic* 12(4), 2011.
3. M. Bojańczyk, B. Klin and S. Lasota, Automata theory in nominal sets, *Logical Methods In Computer Science* 10(3:4), 1–44, 2014.
4. B. Bollig, P. Habermehl, M. Leucker and B. Monmege, A fresh approach to learning register automata, 17th International Conference on Developments in Language Theory (DLT 2013), LNCS 7907, 118–130.
5. S. Cassel, F. Howar, B. Jonsson and B. Steffen, Active learning for extended finite state machines, *Formal Aspects of Computing* 28, 233–263, 2016.
6. E. Y. C. Cheng and M. Kaminski, Context-free languages over infinite alphabets, *Acta Informatica* 35, 245–267, 1998.
7. E. M. Clarke, Jr., O. Grumberg, D. Kroening, D. Peled and H. Veith, *Model Checking*, Second. Edition, Chapter 15: Verification with Automata Learning, The MIT Press, 2018.
8. J. M. Cobleigh, D. Giannakopoulou and C. S. Păsăreanu, Learning assumptions for compositional verification, 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003), LNCS 2619, 331–346.
9. S. Demri and R. Lazić, LTL with the freeze quantifier and register automata, *ACM Transactions on Computational Logic* 10(3), 2009.
10. D. Figueira and L. Segoufin, Bottom-up automata on data trees and vertical XPath, 28th Symposium on Theoretical Aspects of Computer Science (STACS 2011), 93–104.
11. M. Gabbay and A. M. Pitts, A new approach to abstract syntax with variable binding, *Formal Aspects of Computing* 13, 341–363, 2002.
12. M. Kaminski and N. Francez, Finite-memory automata, *Theoretical Computer Science* 134, 329–363, 1994.
13. M. Kaminski and T. Tan, Tree automata over infinite alphabets, *Pillars of Computer Science 2008*, LNCS 4800, 386–423.
14. M. Leucker, Learning meets verification, 5th International Symposium on Formal Methods for Components and Objects (FMCO 2006), LNCS 4709, 127–151.
15. L. Libkin, T. Tan and D. Vrgoč, Regular expressions for data words, *Journal of Computer and System Sciences* 81(7), 1278–1297, 2015.
16. L. Libkin and D. Vrgoč, Regular path queries on graphs with data, 15th International Conference on Database Theory (ICDT 2012), 74–85.
17. J. Moerman, M. Sammartino, A. Silva, B. Klin and M. Szynwelski, Learning nominal automata, 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017), 613–625.
18. A. S. Murawski, S. J. Ramsay and N. Tzevelekos: Reachability in pushdown register automata, *J. Computer and System Sciences* 87, 58–83, 2017.
19. F. Neven, T. Schwentick and V. Vianu, Finite state machines for strings over infinite alphabets, *ACM Trans. on Computational Logic* 5(3), 403–435, 2004.
20. D. Peled, M. Y. Vardi and M. Yannakakis, Black box checking, *Formal Methods for Protocol Engineering and Distributed Systems (FORTE/PSTV 1999)*, 225–240.
21. Y. Sakakibara, Learning context-free grammars from structural data in polynomial time, *Theoretical Computer Science* 76, 223–242, 1990.

- 22. H. Sakamoto and D. Ikeda, Intractability of decision problems for finite-memory automata, *Theoretical Computer Science* 231, 297–308, 2000.
- 23. R. Senda, Y. Takata and H. Seki, Complexity results on register context-free grammars and register tree automata, 15th International Colloquium on Theoretical Aspects of Computing (ICTAC 2018), LNCS 11187, 415–434.
- 24. R. Senda, Y. Takata and H. Seki, Reactive synthesis from visibly register pushdown automata, 18th International Colloquium on Theoretical Aspects of Computing (ICTAC 2021), LNCS 12819, 334–353.

Appendix

Proposition 2. *Let $X = \llbracket C, S \rrbracket$ and $Y = \llbracket D, T \rrbracket$ be single orbit nominal sets. For every equivariant function $f : X \rightarrow Y$, there is an injection u from D to C satisfying $uS \subseteq Tu$ and $f([\pi|_C]_S) = [u]_T \cdot \pi$, for all $\pi \in G$, where $uS = \{us \mid s \in S\}$ and $Tu = \{tu \mid t \in T\}$. Conversely, for every injection $u : D \rightarrow C$ satisfying $uS \subseteq Tu$, $f([\pi|_C]_S) = [u]_T \cdot \pi$ is an equivariant function from X to Y .*

Proof. Let $f : X \rightarrow Y$ be an equivariant function. Let u be an arbitrary element of the equivalent class $f([e|_C]_S)$; i.e., $f([e|_C]_S) = [u]_T$. By [3, Proposition 9.16], we can assume the type of u is $u : D \rightarrow C$ without loss of generality. Then, $f([\pi|_C]_S) = f([e|_C]_S \cdot \pi) = f([e|_C]_S) \cdot \pi = [u]_T \cdot \pi$ for all $\pi \in G$ because f is equivariant. For any $x \in X$ and $\pi \in G$, if $x \cdot \pi = x$, then $f(x) \cdot \pi = f(x \cdot \pi) = f(x)$ because f is equivariant. Therefore,

$$\begin{aligned} \forall \pi \in G. [e|_C]_S \cdot \pi &= [e|_C]_S \Rightarrow [u]_T \cdot \pi = [u]_T \\ \forall \pi \in G. [\pi|_C]_S &= [e|_C]_S \Rightarrow [u\pi]_T = [u]_T \\ \forall \pi \in G. \pi|_C &\equiv_S e|_C \Rightarrow u\pi \equiv_T u \\ \forall \pi \in G. \pi|_C (e|_C)^{-1} &\in S \Rightarrow u\pi u^{-1} \in T \\ \forall \pi \in G. \pi|_C &\in S \Rightarrow u\pi u^{-1} \in T \\ \forall \tau \in S. u\tau u^{-1} &\in T \\ \forall \tau \in S. \tau &\in u^{-1}Tu \\ S &\leq u^{-1}Tu \\ uS &\subseteq Tu. \end{aligned}$$

Let $u : D \rightarrow C$ be an injective function satisfying $uS \subseteq Tu$. We prove that $f([\pi|_C]_S) = [u]_T \cdot \pi$ is an equivariant function from X to Y . We first show that f is well-defined. When $[\pi|_C]_S = [\sigma|_C]_S$,

$$\begin{aligned} \pi|_C &\equiv_S \sigma|_C \\ (\pi|_C)(\sigma|_C)^{-1} &\in S \\ u(\pi|_C)(\sigma|_C)^{-1} &\in uS \subseteq Tu \\ u(\pi|_C)(\sigma|_C)^{-1}u^{-1} &\in T \\ (u\pi)|_D (u\sigma)|_D^{-1} &\in T \\ (u\pi)|_D &\equiv_T (u\sigma)|_D \\ [(u\pi)|_D]_T &= [(u\sigma)|_D]_T \\ [u]_T \cdot \pi &= [u]_T \cdot \sigma \end{aligned}$$

Moreover, f is equivariant because $f([\pi|_C]_S \cdot \rho) = f([(\pi\rho)|_C]_S) = [u]_T \cdot (\pi\rho) = ([u]_T \cdot \pi) \cdot \rho = f([\pi|_C]_S) \cdot \rho$. \square

Lemma 11. *If $(S, \mathcal{E}, \mathcal{T})$ and $(S, \mathcal{E}', \mathcal{T}')$ are observation tables such that $\mathcal{E} \subsetneq \mathcal{E}'$ and $\mathcal{T}'(e[s]) = \mathcal{T}(e[s])$ for all $e \in \mathcal{E}$ and $s \in S$, then $\text{row}_{(S, \mathcal{E}, \mathcal{T})}(S) \prec \text{row}_{(S, \mathcal{E}', \mathcal{T}')} (S)$.*

Proof. For readability, let row denote $\text{row}_{(\mathcal{S}, \mathcal{E}, \mathcal{T})}$ and row' denote $\text{row}_{(\mathcal{S}, \mathcal{E}', \mathcal{T}')}$. By the definition of \prec , it suffices to show that there is an equivariant surjection from a subset of $\text{row}'(\mathcal{S})$ to $\text{row}(\mathcal{S})$ and they are not isomorphic. We first show that the function $h : \text{row}'(\mathcal{S}) \rightarrow \text{row}(\mathcal{S})$ defined by $\text{row}'(s) \mapsto \text{row}(s)$ is surjective and equivariant. By the following, h is well-defined:

$$\begin{aligned} \text{row}'(s_1) = \text{row}'(s_2) &\Leftrightarrow \forall e \in \mathcal{E}' . \mathcal{T}'(e[s_1]) = \mathcal{T}'(e[s_2]) \\ &\Rightarrow \forall e \in \mathcal{E} . \mathcal{T}(e[s_1]) = \mathcal{T}(e[s_2]) \\ &\Leftrightarrow \text{row}(s_1) = \text{row}(s_2). \end{aligned}$$

Moreover, h is equivariant by $\text{row}'(s) \cdot \pi = \text{row}'(s \cdot \pi) \mapsto \text{row}(s \cdot \pi) = \text{row}(s) \cdot \pi$. It is easy to see that h is surjective.

Next, we show that $\text{row}(\mathcal{S})$ and $\text{row}'(\mathcal{S})$ are not isomorphic. To show this, we only have to show that

$$\text{row}(\text{Orbit}(s_1)) (= \text{row}(\text{Orbit}(s_2))) \prec \text{row}'(\text{Orbit}(s_1)) \cup \text{row}'(\text{Orbit}(s_2)) \quad (1)$$

for $s_1, s_2 \in \mathcal{S}$ such that $\text{row}(s_1) = \text{row}(s_2)$ and $\text{row}'(s_1) \neq \text{row}'(s_2)$ because $\text{row}(\mathcal{S})$ and $\text{row}'(\mathcal{S})$ are orbit finite sets. Let h' be h where the domain is restricted to $\text{row}'(\text{Orbit}(s_1)) \cup \text{row}'(\text{Orbit}(s_2))$. It is easy to see that h' is a surjective and equivariant function from $\text{row}'(\text{Orbit}(s_1)) \cup \text{row}'(\text{Orbit}(s_2))$ to $\text{row}(\text{Orbit}(s_1))$. We show that $\text{row}'(\text{Orbit}(s_1)) \cup \text{row}'(\text{Orbit}(s_2))$ is not isomorphic to $\text{row}(\text{Orbit}(s_1))$. If $\text{row}'(\text{Orbit}(s_1)) \neq \text{row}'(\text{Orbit}(s_2))$ then this holds trivially. Assume that $\text{row}'(\text{Orbit}(s_1)) = \text{row}'(\text{Orbit}(s_2))$ and let

$$\begin{aligned} \llbracket C, S \rrbracket &= \text{row}(\text{Orbit}(s_1)), \text{ and} \\ \llbracket D, T \rrbracket &= \text{row}'(\text{Orbit}(s_1)) \end{aligned}$$

by Proposition 1. By the definition of $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$, $\text{row}'(s_1) = [\pi_1|_D]_T$ and $\text{row}'(s_2) = [\pi_2|_D]_T$ for some $\pi_1, \pi_2 \in G$. By Proposition 2, we can write h' as $h'([\pi|_D]_T) = [\sigma|_C]_S \cdot \pi$ for some $\sigma \in G$. Thus,

$$\begin{aligned} h'(\text{row}'(s_1)) &= h([\pi_1|_D]_T) = [\sigma|_C]_S \cdot \pi_1 = [(\sigma\pi_1)|_C]_S = \text{row}(s_1), \\ h'(\text{row}'(s_2)) &= h([\pi_2|_D]_T) = [\sigma|_C]_S \cdot \pi_2 = [(\sigma\pi_2)|_C]_S = \text{row}(s_2). \end{aligned}$$

Assume that $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are isomorphic, i.e., there exists an equivariant bijection $f : \llbracket C, S \rrbracket \rightarrow \llbracket D, T \rrbracket$. By Proposition 2, f can be written as $f([\pi|_C]_S) =$

$[\rho|_D]_T \cdot \pi$ for some $\rho \in G$. We have

$$\begin{aligned}
\text{row}(s_1) &= \text{row}(s_2) \\
f(\text{row}(s_1)) &= f(\text{row}(s_2)) \\
f([\sigma\pi_1|_C]_S) &= f([\sigma\pi_2|_C]_S) \\
[\rho|_D]_T \cdot (\sigma\pi_1) &= [\rho|_D]_T \cdot (\sigma\pi_2) \\
[(\rho\sigma\pi_1)|_D]_T &= [(\rho\sigma\pi_2)|_D]_T \\
(\rho\sigma\pi_1)|_D &\equiv_T (\rho\sigma\pi_2)|_D \\
(\rho\sigma\pi_1)|_D (\rho\sigma\pi_2)|_D^{-1} &\in T \\
(\rho\sigma\pi_1\pi_2^{-1}\sigma^{-1}\rho^{-1})|_D &\in T \\
(\rho\sigma\pi_1\pi_2^{-1})|_D &\in T\rho\sigma.
\end{aligned}$$

Because f is an equivariant and bijective function, $\rho|_D S = T\rho|_D$ by the proof of Lemma 1. Thus, we have $|S| = |T|$. By Proposition 2 and h' , $\sigma|_C T \subseteq S\sigma|_C$. By $\sigma|_C T \subseteq S\sigma|_C$ and $|S| = |T|$, $\sigma|_C T = S\sigma|_C$. By $\rho|_D S = T\rho|_D$ and $\sigma|_C T = S\sigma|_C$, $\rho|_D \sigma|_C T = T\rho|_D \sigma|_C$. Thus,

$$\begin{aligned}
(\rho\sigma\pi_1\pi_2^{-1})|_D &\in T\rho\sigma = \rho\sigma T \\
(\pi_1\pi_2^{-1})|_D &\in T \\
\pi_1|_D &\equiv_T \pi_2|_D \\
[\pi_1|_D]_T &= [\pi_2|_D]_T \\
\text{row}'(s_1) &= \text{row}'(s_2).
\end{aligned}$$

This is a contradiction, and hence $\llbracket C, S \rrbracket$ and $\llbracket D, T \rrbracket$ are not isomorphic. \square