Tree-Interpretable Linear Orders

Studythesis Eduard Kamburjan
Advisor Priv.-Doz. Dr. Achim Blumensath

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Contents

Introduction

The study of infinite structures involves to algorithmically determine properties of these structures. To process such structures one wants to represent them in a finite way that allows the construction of effective algorithms. This raises the question of which finite representations can be used for this purpose and which infinite structures can be represented with them. In this work we present five representations and show that they describe the same set of linear orders.

Our basis are linear orders which are MSO-interpretable in the complete binary tree, a well understood structure with decidable MSO-theory [ER66]. The original motivation for some of the representations was to find operations that preserve the decidability of the theory of the complete binary tree while constructing more sophisticated graphs. We only consider linear orders to show the main points of the representations and to give complete but short equivalence proofs.

We divide the representations into two groups: *order terms* and *systems of order equations* can be seen as concepts that *generate* graphs, while *inverse substitutions* and *prefix-recognizability* describe how a graph can be *accepted*. This is analogous to the generator/acceptor view on regular grammars and regular expressions or finite automata in theoretical computer science and is not supposed to imply that a representations can not be used to fulfill the task of the other group.

Outline

This work is organized as follows: In Section 1 we review basic definitions. In Section 2 we introduce the concepts of acceptors and show their equivalence. In Section 3 we introduce the concepts of generators and show their equivalence. We conclude in Section 4 with a brief overview of generalizations to larger graph classes and further representations for them.

1 Fundamentals

In this work we consider mostly monadic second-order logic and directed graphs, thus we review the basic definitions of them.

1.1 Graphs and Trees

Let C be a finite set.

- A *C-colored graph* is a tuple $\mathfrak{G} = (V, E, (P_c)_{c \in C})$ where V is called the vertex-set, $E \subseteq V \times V$ the edge-set and each $P_c \subseteq V$ a color.
- A *path* from v_0 is a sequence $(v_0, v_1, \dots)$ such that $\forall i.(v_i, v_{i+1}) \in E$ and all vertices but the endpoints are distinct: $(\forall 0 < i, j)\big[i \neq j \to v_i \neq v_j\big]$
- A path $p = (v_0, v_1, \dots)$ from v_0 is a *cycle* if p has length $i \neq 0$ and $v_0 = v_{i-1}$ holds.
- A graph forms a *tree* if it contains no cycle, and there is a root vertex r such that for every vertex $v \neq r$ there is a unique path from r containing v.
- A graph forms a *linear order* if E has the following properties:
  - Reflexivity: $(\forall v \in V)\big[(v, v) \in E\big]$
  - Transitivity: $(\forall v, w, u \in V)\big[((v, w) \in E \wedge (w, u) \in E) \to (v, u) \in E\big]$
  - Totality: $(\forall v, w \in V)\big[(v, w) \in E \vee (w, v) \in E\big]$

  In this case we denote E by $\leq$.

Definition
Let $L = (V, \leq)$ be a linear order and $L_1, \dots, L_n$ suborders of L. The suborders $L_1, \dots, L_n$ are *mutually dense* if for every suborder L_k, every $j \neq k$ and every two vertices $v, w \in L_k$ with $v < w$, we can find a vertex u with $v < u < w$ that is element of L_j.

We denote the uncolored linear order on the natural numbers with $\omega = (\mathbb{N}, \leq)$ and its reverse with $-\omega = (\mathbb{Z} \setminus \mathbb{N}, \leq)$. Furthermore, we denote the last k elements of a finite order L by $\mathit{suffix}_k(L)$.

The simplest operation on graphs is restriction to a set, which removes all vertices that are not in a given set.

Definition (Restriction)
Given a graph $\mathfrak{G} = (V, E, (P_c)_{c \in C})$ and a set S, the *restriction* of $\mathfrak{G}$ to S is

$$\mathfrak{G} \restriction S := \big(V \cap S, E \cap S \times S, (P_c \cap S)_{c \in C}\big)$$

We are mostly concerned with linear orders and trees, especially complete trees.

Definition (Complete Tree)
Let $n \in \mathbb{N}$. A tree $\mathfrak{T}_n = (V, E)$ is *n-complete* if its universe V and its edge relation E have this form:

$$V = \{0, \dots, n-1\}^*$$
$$E = \{(v, w) \mid v \preceq w \wedge |v| = |w| + 1\}$$

where $\preceq$ is the usual prefix relation on finite strings.

For simplicity we additionally define the successor functions $next_n(v, v') \iff v' = vn$ and the left-to-right order for $n = 2$:

$$v \leq_{\mathfrak{T}_2} v' \iff \big(\exists w \in V\big)\big(\exists k, i \in \{0,1\}\big) \big[(wk \preceq v \wedge wi \preceq v' \wedge k < i) \vee (v'0 \preceq v) \vee (v1 \preceq v')\big]$$

In the case $n = 2$ we denote $next_0$ with *left* and $next_1$ with *right*.

Lemma 1
The complete binary tree $\mathfrak{T}_2$ is dense with respect to $\leq_{\mathfrak{T}_2}$.

Proof. Let v, v' be two different vertices with $v \leq_{\mathfrak{T}_2} v'$. If we can reach v from v' with a path $v', w, \dots, v$, we may assume w.l.o.g. $w = left(v')$. In this case $v <_{\mathfrak{T}_2} right(v) <_{\mathfrak{T}_2} v'$.

If there is no path between the vertices, let v'' be the longest prefix of v' and v and by definition of $\leq_{\mathfrak{T}_2}$ the formula $v <_{\mathfrak{T}_2} v'' <_{\mathfrak{T}_2} v'$ holds. □

We need the following lemma to extend mutually dense orders to mutually dense subsets of $\mathfrak{T}_2$.

Lemma 2
For every $n \in \mathbb{N}$ we can find n mutually dense sets $P_1, \dots, P_n \subseteq \{0,1\}^*$. Furthermore we can formulate for each such set P_k a MSO-formula $\phi_k(x)$ that holds iff x is in P_k.

Proof. Set $P_k := \{w \mid suffix_{k+1}(w) = 10^k\}$. We show that $P_1, \dots, P_n$ are mutually dense in $\leq_{\mathfrak{T}_2}$. Let $v, w \in P_k$ with $v \leq_{\mathfrak{T}_2} w$ for a fixed k, and $j \neq k$.

- Case $v \preceq w$: In this case $v = u10^k$ and $w = vx10^k = u10^k x10^k$ for some x. For every j the subtree under $vx0$ lies between v and w: $u10^k \leq_{\mathfrak{T}_2} vx010^j \leq_{\mathfrak{T}_2} u10^k x10^k$.
- Case $v \not\preceq w$: In this case $w = x10^k$ for some x and the subtree under $x0$ is between v and w. Thus we construct: $v \leq_{\mathfrak{T}_2} x010^j \leq_{\mathfrak{T}_2} w$

We set $\phi_k(x) := \exists y.\big[x = right\big(\underbrace{left(left(\dots}_{k \text{ times}}(y)\dots)\big)\big]$ □

Note that this is no partition. We slightly extend these sets to obtain subtrees, which contain no element of any of these sets:

Lemma 3
For every $n \in \mathbb{N}$ we can find n mutually dense sets $Q_1, \ldots, Q_n \subseteq \{0,1\}^*$ such that the right subtree under any element of any of these sets contains no element from these sets. Furthermore we can formulate for each such set Q_k a MSO-formula $\psi_k(x)$ that defines Q_k.

Proof. We stretch the sets from Lemma 2 by putting in 0's at every second position in the words of P_k:

$$Q_k := \{w \mid w = w_0 0 w_1 0 \ldots 0 w_n \text{ and } w_0 w_1 w_n \cdots \in P_k \text{ for some } n \in \mathbb{N} \text{ and } w_0, \ldots, w_n \in \{0,1\}\}$$

In every word every character at an odd position is a 0, thus the right subtree of any element contains no further element as it has a prefix of odd length that ends with a 1. The claim of mutually density is analogous to Lemma 2 and we set:

$$\psi_k(x) := even(x) \wedge \forall x'.[x' \leq x \wedge even(x') \Rightarrow \exists z.[x' = left(z)]] \wedge \exists y.[x = right(\underbrace{left(left(\ldots}_{2k \text{ times}}(y)\ldots))))]$$

Where $even(x)$ is a shortcut for

$$\forall X.[X(\epsilon) \wedge \forall z.[X(z) \Rightarrow X(left(left(z)) \wedge X(left(right(z)) \wedge X(right(left(z)) \wedge X(right(right(z))] \Rightarrow X(x)]$$

which denotes that the length of the path to an element is even. □

To constrain the evaluation of a formula to a certain subtree we introduce *restriction*.

Definition
Given a formula ϕ and a node p in $\mathfrak{T}_2$ the *restriction* of ϕ to the subtree under p is obtained by replacing every subformula of the form $\exists x.\psi$ with $\exists x.[p \preceq x \wedge \psi]$ and every subformula of the form $\forall x.\psi$ with $\forall x.[p \preceq x \rightarrow \psi]$. We denote the resulting formula with $\phi^{(p)}$.

For readability we introduce the following shortcuts:

Definition
Given a node v in $\mathfrak{T}_2$ we write $\succ v$ for the left subtree under v and $\prec v$ for the right subtree under v

Also we use define MSO-formulas *RBranch* and *LBranch* defining the right-most and left-most branch of the complete binary tree.

$$RBranch(v) := \exists B.[B(\epsilon) \wedge \forall x.[B(x) \leftrightarrow B(right(x))] \wedge \forall y.\neg B(left(y)) \wedge B(v)]$$
$$LBranch(v) := \exists B.[B(\epsilon) \wedge \forall x.[B(x) \leftrightarrow B(left(x))] \wedge \forall y.\neg B(right(y)) \wedge B(v)]$$

1.2 Operations on Graphs

We shortly review the definitions of sums and products of C-colored orders with a finite color set C: The sum $L \oplus L'$ is the the result of the disjoint union of L and L', and placing all elements of L before the elements L':

$$(V, \leq, (P_c)_{c\in C}) \oplus (V', \leq', (P'_c)_{c\in C}) \;=\; (V \dot\cup V', \leq \cup \leq' \cup V \times V', (P_c \cup P'_c)_{c\in C})$$

The product $L \otimes L'$ is the result of replacing every element of L' by a copy of L and maintaining the coloring of L:

$$(V, E, (P_c)_{c\in C}) \times (V', E', (P'_c)_{c\in C}) \;=\; (V \times V', \leq', (P''_c)_{c\in C})$$

with $(v_1, v_2) \leq (v'_1, v'_2) \iff (v_2, v'_2) \in E' \vee \left[v'_2 = v_2 \wedge (v_1, v'_1) \in E \right]$ and
$(v_1, v_2) \in P''_c \iff v_1 \in P_c$.

An operation for generating different dense orders is σ: given $n \in \mathbb{N}$ linear orders $L_1, \ldots, L_n$ their *shuffle* is obtained from a partition of $\mathbb{Q}$ into n mutually dense subsets $A_1, \ldots, A_n$ and replacing each element of A_i with a copy of L_i.

We need the following statement for the well-definedness of the shuffle:

Lemma 4
For every $n \in \mathbb{N}$ we can partition $\mathbb{Q}$ into n mutually dense subsets and for all such partitions the resulting shuffles are isomorphic.

A proof can be found in [Ros82].

An MSO-interpretation is a way to extract elements from a structure by defining a predicate which the elements must satisfy in the source structure and defining new relation by similar predicates.

Definition
Let τ, τ' be two relational signatures. A *MSO-interpretation* from a τ-structure $\mathfrak{A}$ into a τ'-structure $\mathfrak{B}$ is a list of MSO-formulas over τ

$$\langle \delta(x), (\phi_r(\overline{y}))_{r\in\tau'} \rangle$$

Where $\overline{y}$ has the arity of the respective r.

Given a τ-structure $\mathfrak{A} = (A, (R_r)_{r\in\tau})$ the result of applying an interpretation $\mathscr{I} = \langle \delta(x), (\phi_r(y))_{r\in\tau'} \rangle$ on $\mathfrak{A}$ is the τ'-structure $\mathfrak{B}$:

$$\mathfrak{B} = \left(\{a \in A \mid \mathfrak{A} \models \delta(a)\}, (\{\overline{y} \in A^{|y|} \mid \mathfrak{A} \models \phi_r(y)\})_{r\in\tau'} \right)$$

We say that $\mathfrak{B}$ is interpreted in $\mathfrak{A}$.

In this work we only consider interpretations from $\mathfrak{T}_2$ into linear orders with a finite set of colors. We emphasize that an interpretation is in this case already a finite representation of an infinite structure, as we have a finite list of finite formulas. However algorithms can make little use of this representation to extract additional information about the linear order.

Given a linear order interpreted in $\mathfrak{T}_2$ the order must not be a suborder of $\leq_{\mathfrak{T}_2}$, however it is always isomorphic to one:

Lemma 5
Let $\mathfrak{A} = (V, \leq, (P_c)_{c \in C})$ be a linear order interpreted in $\mathfrak{T}_2$. Than there is a linear order $\mathfrak{A}' = (V', \leq_{\mathfrak{T}_2}, (P'_c)_{c \in C})$ that is isomorphic to $\mathfrak{A}$ and is interpreted in $\mathfrak{T}_2$.

Proof. We first observe that giving two distinct subtrees t_1, t_2 there can be only finitely many jumps (adjacent pairs $e_1 \leq e_2$ with $e_1 \in t_1, e_2 \in t_2$ or vice versa.) between them in a linear order interpreted in $\mathfrak{T}_2$ and we can retrieve a bound k on the number from the structure of the interpretation [Blu03].

We construct a function $i : V \to V'$ for a given node inductively by splitting the tree above each node and counting how many jumps are traversed before one reaches this node and ordering the nodes by the binary encoding of the jump count and the original node.

The following function assigns the jump counts under a given node v before another node w:

$$count : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$$

$$count(v,w) = \left| \left\{ (v_1, v_2) \in V \mid v_1 \in \succ v \wedge v_2 \in \prec v \wedge v_1, v_2 \leq_{\mathfrak{T}_2} w \wedge \forall v'. \left[v' \leq v_1, v_2 \vee v' \geq v_1, v_2 \right] \right\} \right| + \left| \left\{ (v_1, v_2) \in V \mid v_1 \in \prec v \wedge v_2 \in \succ v \wedge v_1, v_2 \leq_{\mathfrak{T}_2} w \wedge \forall v'. \left[v' \leq v_1, v_2 \vee v' \geq v_1, v_2 \right] \right\} \right|$$

The first part of the formulas describing the sets ensures that (v_1, v_2) are in different subtrees and the second part that there is no v' between them.

Furthermore, when we split at a node v and count the jumps we need to remember where v is ordered, by checking whether v is before or after a given node w:

$$mid(v,w) = \begin{cases} 0 & \text{if } v > w \\ 1 & \text{otherwise} \end{cases}$$

The *flat* function saves this and the counting information for each position:

$$flat : \{0,1\}^* \times \mathbb{N} \times \{0,1\}^* \to \{0,1\} \times \{0,\ldots,k\} \times \{0,1\}^*$$

$$flat(v,i,w) = \Big(mid(v, w_0 \ldots w_i), count(v, w_0 \ldots w_i), w_i \Big) \circ flat(v w_i, i+1, w) \text{ if } i < |w|$$

$$flat(v,i,w) = \Big(\epsilon, \epsilon, \epsilon \Big) \text{ otherwise}$$

Where $\circ$ is componentwise concatenation. The isomorphism is given by assigning to every vertex $v \in V$ the binary encoding of $flat(\epsilon, 0, v)$. □

1.3 Regular Languages

Regular languages are a well-understood class of languages which we use to represent regular structure in graphs. We denote the set of all regular languages over an alphabet Σ with $\mathsf{Reg}(\Sigma)$. The following result is the base for their study[EB58]:

Theorem 1
The following statements are equivalent:

- L is regular
- There is a deterministic finite automaton A with $L(A) = L$
- There is a MSO-formula ϕ with $w \in L \iff w \models \phi$
- There is a MSO-formula $\phi_L(x)$ with $w \in L \iff \mathfrak{T}_2 \models \phi(w)$

We call ϕ_L the formula associated with L.

Furthermore we can use formulas to characterize suffixes:

Lemma 6
Let L be a regular language over Σ. Then there is a formula $\psi_L(x, y)$ such that

$$\mathfrak{T}_2 \models \psi_L(x, y) \iff (\exists z \in \Sigma^*)\big[y = xz \text{ and } z \in L\big]$$

We call ψ_L also associated with L as it is distinguishable from ϕ_L by its number of variables.

Regular languages are closed under several common operations, including union, intersection, negation and reversion. In particular they are closed under concatenation and we denote the concatenation of two languages A, B with $A \circ B$.

We additionally need the following property to deal with paths over graphs:

Lemma 7
Let L be a regular language over the alphabet Σ, and $N \subseteq \Sigma$ a subset of the alphabet. The set of all words in L which first contain only elements of N and then only elements of $\Sigma \setminus N$ is a finite union of concatenated pairs of regular languages, e.g. the following formula holds:

$$\big(\exists n \in \mathbb{N}\big)\big(\exists A_1, ..., A_n \in \mathsf{Reg}(N)\big)\big(\exists B_1, ..., B_n \in \mathsf{Reg}(\Sigma \setminus N)\big)$$
$$\Big[L \cap N^*(\Sigma \setminus N)^* = \bigcup_{i \leq n} A_i \circ B_i\Big]$$

Proof. As L is regular there is a deterministic finite automaton $A = \big(\Sigma, Q, q_0, \Delta, F\big)$ with $L(A) = L$ and $Q = \{q_0, ..., q_n\}$. We can restrict this automaton in such a way that it only works on a subset of the alphabet by deleting all edges that are labelled with a letter from the complement of this subset. This gives rise to two automata which work on N and $\Sigma \setminus N$:

$$A^N := \big(N, Q, q_0, \Delta^N, F\big)$$
$$A^{-N} := \big(\Sigma \setminus N, Q', q_0', \Delta^{-N}, F'\big)$$

where $Q' = \{q'_0, \dots, q'_n\}$ is a disjoint copy of Q and Δ^N, Δ^{-N} are the restrictions of Δ to the respective state set and input alphabets:

$$\Delta^N = \Delta \cap Q \times N \times Q$$
$$\Delta^{-N} = \Delta \cap Q' \times \Sigma \setminus N \times Q'$$

We combine these automata to one that first works on elements of N and then on those of $\Sigma \setminus N$ by adding ε edges between a state of A^N and its counterpart in A^{-N}:

$$A^\Sigma := \left(\Sigma, Q \cup Q', q_0, \Delta^N \cup \Delta^{-N} \cup \{(q_i, \varepsilon, q'_i) \in Q \times \{\varepsilon\} \times Q'\}, F'\right)$$

The language $L(A^\Sigma)$ is the required restriction of L to $N^*(\Sigma \setminus N)^*$, it remains to show that we can split $L(A^\Sigma)$ into a finite union.

For this we split A^Σ into subautomata according to the newly introduced ε-edges. For every state q_i we restrict the state set to the states that can reach q_i and thus accepts the first part of the input, and to those that are reachable from q'_i. If no final state is reachable or this edge is not reachable from the start state we set $A_{\overline{q_i}}$ to the automata that never accepts and otherwise:

$$A_{\overline{q_i}} := \left(\Sigma, \{q \in Q \cup Q' | reach(q, q_i) \vee reach(q'_i, q)\}, q_0, \Delta^\Sigma \restriction Q_{\overline{q_i}} \times \Sigma \cup \{\varepsilon\} \times Q_{\overline{q_i}}, F'\right)$$

As A is finite, A^Σ is finite and there are only finitely many such automata. Each of these automata consists of two parts which are connected by an ε-edge. If we split these automata again along this edge we get the required languages A_i, B_i:

$$A^N_{\overline{q_i}} := \left(N, \{q \in Q | reach(q, q_i)\}, q_0, \Delta^\Sigma \cap Q \times N \cup \{\varepsilon\} \times Q, \{q_i\}\right)$$

$$A^{-N}_{\overline{q_i}} := \left(\Sigma \setminus N, \{q \in Q' | reach(q'_i, q)\}, q'_i, \Delta^\Sigma \cap Q' \times \Sigma \setminus N \cup \{\varepsilon\} \times Q', F'\right)$$

For every q_i whose automaton always rejects we set $A_i = B_i = \emptyset$, otherwise we set $A_i := L(A^N_{\overline{q_i}})$ and $B_i := L(A^{-N}_{\overline{q_i}})$. □

2 Representations of Acceptors

The representations of acceptors considered in the next chapter differ from the generators in the aspect that they only represent the structure of the linear ordering in the binary tree and not the coloring.

To use them for the same class of graphs, we identify colored graphs $\mathfrak{G} = (V, E, (P_c)_{c \in C})$ with the pair $\left((V, E), (P_c)_{c \in C}\right)$. This requires a finite representation of the colors and it suffices for this work to consider regular colorings.

The representations of acceptors define the ordering between two vertices by their relative position and the path between them in the complete binary tree, thus extracting the order from this structure. To describe the relation between two vertices both approaches describe vertices with regular languages and reduce the paths between them to paths which are labelled by a regular language.

2.1 Inverse Substitutions

Inverse substitutions extract a graph from a tree by specifying the edge relation through a homomorphism from their label in the extracted graph to paths.

As in the complete tree all edges are pointed away from the root, we introduce mirrored trees to enable paths between arbitrary vertices.

Definition (Mirrored Trees)
Given a complete tree $\mathfrak{T}_n = \big(\{0,\dots,n-1\}^*, E\big)$ we define the mirrored complete tree $\overline{\mathfrak{T}_n} = (V, E')$ by adding all inverted all edges:

$$E' := E \cup \{(u,v) | (v,u) \in E\}$$

To talk about labelled paths we regard an edge $(v,w) \in E'$ as labelled by n, if $w = vn$ and as labelled by $\overline{n}$ if $v = wn$. Thus if we descend from a vertex v to vn this edge is labelled by n and the mirror of this edge by $\overline{n}$. We write that an edge between v,w is labelled by n in a graph G as $v \to_n^G w$.

We say that a path $(v_0, v_1, ..., v_n) \in V^*$ from v_0 to v_n is labelled by $a = a_0 a_1 ... a_{n-1} \in \{0, \overline{0}, ..., n-1, \overline{n-1}\}^*$ if for each $i < n$ the edge $(v_i, v_{i+1}) \in E'$ is labelled by a_i. We write that there is a path between v,w that is labelled by a in a graph G as $v \Rightarrow_a^G w$.

Definition (Substitution)
Let R be a finite set. An extended substitution h is a homomorphism from words in R^* into the set of languages over $\{0, \overline{0}, ..., n-1, \overline{n-1}\}^*$:

$$h : R^* \to \mathscr{P}(\{0, \overline{0}, ..., n-1, \overline{n-1}\}^*)$$

We say that h is regular if for every $r \in R$ the language $h(r)$ is regular. A graph $\mathfrak{G} = (V,E)$ is an inverse regular substitution of h on $\mathfrak{T}_n = (V, E', (P_c)_{c \in C})$ if for each $v,w \in V$ any edge between them that is labelled by $a \in R$ there is a word $w \in \{0, \overline{0}, ..., n-1, \overline{n-1}\}^*$ in $h(a)$ that labels a path between v and w in $\overline{\mathfrak{T}_n}$:

$$v \to_a^{\mathfrak{G}} w \iff (\exists\, u \in \{0, \overline{0}, ..., n-1, \overline{n-1}\}^*).\ \Big[u \in h(a) \wedge v \Rightarrow_u^{\overline{\mathfrak{T}_n}} w\Big]$$

If the graph has this form we write $\mathfrak{G} = h^{-1}(\mathfrak{T}_2)$

Note that if w is a path, it may not be the shortest path between its endpoints. The only difference between the shortest path and other paths in the mirrored complete tree is that these contain a subpath that is a loop, i.e. their path label has a subsequence of the form $v\overline{v^{-1}}$ or $\overline{v}v^{-1}$. The following lemma shows that in case of inverse regular substitutions it suffices to consider the shortest paths[Cau96].

Lemma 8
Let h be a homomorphism $h : R \to \{0,1,\overline{0},\overline{1}\}^*$. Then there is a homomorphism $\underline{h} : R \to \{0,1,\overline{0},\overline{1}\}^*$ such that for every $r \in R$ the set $\underline{h}(r)$ contains no words that have a subword of the form $v\overline{v^{-1}}$ or $\overline{v}v^{-1}$ for any $v \in \{0,1\}$ and $h^{-1}(\mathfrak{T}_2) = \underline{h}^{-1}(\mathfrak{T}_2)$.

Proof. $\underline{h}$ is constructed by iteratively removing every subword of the required form. If this word was a path label, the resulting word labels a path between the same vertices but without a detour of v. Note that this is a property of trees, not of general graphs. □

Substitutions are defined on uncolored graphs, we need to encode the coloring as additional edges. To encode a family of regular colors $P = (P_c)_{c\in C}$ in $\mathfrak{T}_2$ we add edges (v, v) which we regard as labelled with c if $v \in c$. We denote such an extended tree with $\mathfrak{T}_2^P$.

Substitutions are a way to extract paths with labels from a regular language from a tree. However to ensure that all vertices form a regular language and to get rid of vertices that are, in some sense, irrelevant one needs to restrict inverse regular substitutions to a regular set. As it turns out with this restriction we can represent all linear orders that are interpreted in the complete binary tree, as the following lemma shows:

Lemma 9
Let $h : \{\sqsubseteq\}^* \to \mathscr{P}(\{0, 1, \overline{0}, \overline{1}\}^*)$ be a regular homomorphism, L a regular language and $Q = (Q_i)_{i\in I}, (P_c)_{c\in C}$ two finite families of regular colors such that $\left(h^{-1}(\mathfrak{T}_2^Q)\restriction_L, (P_c)_{c\in C}\right)$ is a linear order. Then there is an interpretation $\mathscr{I} = \langle \delta(x), \psi_\leq, (\psi_c)_{c\in C}\rangle$ such that $\left(h^{-1}(\mathfrak{T}_2^Q)\restriction_L, (P_c)_{c\in C}\right) = \mathscr{I}(\mathfrak{T}_2)$.

Proof. By Lemma 8 it suffices to regard $\underline{h}$ and by Lemma 7 we can represent any regular language by a union $\cup_{i\leq n} A_i \circ B_i$ for some regular languages $A_0, \ldots, A_n \subseteq \{\overline{0}, \overline{1}\}^*$ and $B_0, \ldots, B_n \subseteq \{0, 1\}^*$. Let $\cup_{i\leq n} A_i \circ B_i$ be such a representation of $h(\sqsubseteq)$. As regular languages are closed under reversing, let ψ_{A_i} be the formula associated with the reverse of A_i where every element $\overline{a}$ is replaced by a, and let ψ_{B_i} be the language associated with B_i.

We modify these formulas to consider the additional information added by the encoded colors Q and set ψ'_{A_i} to the formula that is obtained from replacing every subformula of the form $E_i(z, z')$ for $i \in I$ by $z = z' \wedge Q_i(z)$ in ψ_{A_i}. We set

$$\psi_\leq(x, y) := \exists w.\Big(w = x \sqcap y \wedge \psi'_{A_i}(w, x) \wedge \psi'_{B_i}(w, y)\Big)$$

where $x \sqcap y$ denotes the longest common prefix of x and y. As L and the colors are regular we set δ and ψ_c to the according associated formulas. □

2.2 Prefix-Recognizable Graphs

Prefix-recognizable graphs capture the structure of the linear ordering by describing an edge between two vertices v and w by dissecting the paths from the root to these vertices.

Definition (Prefix-Recognizable Graphs)
A graph $(\mathscr{V}, E, (P_c)_{c\in C})$ is *prefix-recognizable* if $\mathscr{V} \subseteq \{0, 1\}^*$ is a regular language, for every $c \in C$ the set P_c is a regular language and E is a finite union of relations of the form $W(U \times V) := \Big\{(wu, wv) \mid w \in W, v \in V, u \in U\Big\}$ with regular W, V, U.

It is easy too see that this is a finite representation as we can describe regular sets with finite MSO-formulas:

Lemma 10
Let $(V, \leq, (P_c)_{c\in C})$ be a prefix-recognizable linear ordering. Then there is an interpretation $I = \langle \delta, \phi_\leq, (\phi_c)_{c\in C}\rangle$ such that $(V, \leq, (P_c)_{c\in C}) = I(\mathfrak{T}_2)$.

Proof. We set δ to be the formula associated with V and ϕ_c the formula associated with the corresponding P_c. By definition $\leq = \bigcup_{i\leq n} W_i(U_i \times V_i)$ for regular languages $(W_i)_{i\leq n}, (U_i)_{i\leq n}, (V_i)_{i\leq n}$ We can define for each $i \leq n$ the relation $W_i(V_i \times U_i)$ by the following formula

$$\psi_i(x, y) := \exists z.\, \left[\phi_{W_i}(z) \wedge \phi_{U_i}(z, x) \wedge \phi_{V_i}(z, y)\right]$$

where ϕ_L is the formula associated with the regular language L. The union of these relations is translated into a disjunction of the formulas:

$$\phi_\leq(x, y) := \bigvee_{i\leq n} \psi_i(x, y)$$ □

The reverse also holds and we first introduce a further representation of regular languages by tree automata and encoding variables in trees.

Lemma 11
Let $\phi(x_1, x_2)$ be a fixed MSO-formula with two free variables and v, w two vertices of the complete binary tree. There is a tree automaton A such that the formula $\phi(v, w)$ holds iff A accepts the complete binary tree with the following additional labeling:

A vertex u is labelled by $[0, 0]$ if $u \neq v, u \neq w$, by $[1, 0]$ if $u = v, u \neq w$, by $[0, 1]$ if $u \neq v, u = w$ and by $[1, 1]$ if $u = v = w$.

A proof can be found in [Tho96].

In [Blu01] this result is used to reduce tree automata to a triple of normal automata:

Lemma 12
Let $I = \langle \delta, \psi_\leq, (\phi_c)_{c\in C}\rangle$ be a interpretation such that $I(\mathfrak{T}_2)$ is a linear order. Then $I(\mathfrak{T}_2)$ is prefix-recognizable.

Proof. Let $A = (Q, \{0, 1\}, \Delta, q_0, F)$ be the tree automaton associated with $\psi_\leq(x, y)$ according to Lemma 11 and $\mathrm{Occ}(t)$ the set of all labels occuring in the tree t. We classify the states Q according to whether and where they occur on a accepting run of A:

$$Q_\emptyset := \left\{q \in Q \mid \mathrm{Occ}(t) = \{[0, 0]\} \text{ for all trees } t \text{ accepted after } A \text{ was in } q \text{ at the root of } t\right\}$$
$$Q_x := \left\{q \in Q \mid \mathrm{Occ}(t) = \{[0, 0], [1, 0]\} \text{ for all trees } t \text{ accepted after } A \text{ was in } q \text{ at the root of } t\right\}$$
$$Q_y := \left\{q \in Q \mid \mathrm{Occ}(t) = \{[0, 0], [0, 1]\} \text{ for all trees } t \text{ accepted after } A \text{ was in } q \text{ at the root of } t\right\}$$
$$Q_{x,y} := \left\{q \in Q \mid \mathrm{Occ}(t) = \{[0, 0], [1, 0], [0, 1]\} \text{ or } \mathrm{Occ}(t) = \{[0, 0], [1, 1]\} \text{ for all trees } t \text{ accepted after } A \text{ was in } q \text{ at the root of } t\right\}$$

The states in Q_x are the ones that are passed through by the automata on some accepting run on the path from the root to the vertex x. Q_y is analogous for y and $Q_{x,y}$ are the states passed through on the paths to x and y.

Thus to describe the path from the root to x it suffices to use the states from Q_x and we define for each $q \in Q_x$ the automaton $A_q^x = (Q \cup \{q_f\}, \{0,1\}, \Delta_q, q, \{q_f\})$ by

$$\begin{aligned}\Delta_q := \quad & \{(p,0,p') \mid (p,[0,0],p',p_0) \in \Delta, p_0 \in Q_\emptyset\} \\ & \cup \{(p,1,p') \mid (p,[0,0],p_0,p') \in \Delta, p_0 \in Q_\emptyset\} \\ & \cup \{(p,\epsilon,q_f) \mid (p,[1,0],p_0,p') \in \Delta, p_0 \in Q_\emptyset\}\}\end{aligned}$$

where the states in $Q_\emptyset$ mark the direction which is not on the path. The first two subsets describe the way down and the last one the accepting state. Note that A_q^x does not walk the whole path from the root to x, but only the part after the common prefix of the paths to x and y. Automata A_q^y are defined analogous. To describe the path down to the longest common prefix of x, y we use the automaton $A_q^{xy} = (Q \cup \{q_f\}, \{0,1\}, \Delta_q', q_0, \{q\})$ with

$$\begin{aligned}\Delta_q := & \{(p,0,p') \mid (p,[0,0],p',p_0) \in \Delta, p_0 \in Q_\emptyset\} \\ & \{(p,0,p') \mid (p,[0,0],p_0,p') \in \Delta, p_0 \in Q_\emptyset\}\end{aligned}$$

Intuitively we can use any state q to extract the common prefix of x and y which ends at a vertex v in the state q. With this we can use the automaton A_q^x to walk the path from v to x and A_q^y to walk the path from v to y. The prefix can we read by A_q^{xy} and we set $\leq = \bigcup_{q \in Q} L(A_q^{xy})\big(L(A_q^x) \times L(A_q^y)\big)$ □

To show that interpretations, prefix-recognizable graphs and inverse substitutions represent the same set of linear orders it remains to show the following lemma:

Lemma 13
Let $\mathfrak{G} = (V, \leq, (P_c)_{c \in C})$ be a prefix-recognizable linear order. Then there is a regular homomorphism $h : \{\sqsubset\}^* \to \mathscr{P}(\{0,1,\overline{0},\overline{1}\}^*)$, a finite family of regular sets $Q = (Q_i)_{i \leq n}$ and a regular language L such that $\big(h^{-1}(\mathfrak{T}_2^Q)\restriction_L, (P_c)_{c \in C}\big) = \mathfrak{G}$.

Proof. As $\mathfrak{G}$ is prefix-recognizable, V is regular and there is a $n \in \mathbb{N}$ such that $\leq = \cup_{i \leq n} W_i(U_i \times V_i)$. L is given by V as both have the purpose of restricting the vertex set:

$$L := V$$

Let $\overline{U_i}$ denote the reverse language of U_i where additionally in every word every letter a is replaced by $\overline{a}$. Let $\$_1, \ldots, \$_n$ be new letters. We mark the vertices where the topmost point of every path between two vertices in $W_i(U_i \times V_i)$ is with a regular language with $\$_i$. As W_i is regular we can set $Q_i := W_i$ to mark those vertices.

Let Arc_i be the language of all words that are labels on the shortest path from some vertex v to some vertex w such that the label is in $\overline{U_i} \circ \$_i \circ V_i$.

$$\mathsf{Arc}_i := \overline{U_i} \circ \$_i \circ V_i$$

Arc_i is regular because it involves only checking whether a word is the reverse of another and checking the membership to regular sets.

We can now give the homomorphism by checking for a word w whether it is contained in Arc.

$$h(\sqsubseteq) := \bigcup_{i \leq n} \mathsf{Arc}_i$$

It remains to show that $h^{-1}(\mathfrak{T}_2^Q, (P_c)_{c \in C}) \restriction_L = \mathfrak{G}$ holds. Let $\sqsubseteq_h$ be the order relation of $h^{-1}(\mathfrak{T}_2^Q, (P_c)_{c \in C}) \restriction_L$.

Let $v, v' \in V$ be two vertices of the tree.

- $v \sqsubseteq_h v' \Rightarrow v \leq v'$:
  If $v \sqsubseteq_h v'$ then there is a word w such that $w \in h(\sqsubseteq)$, i.e. w is in Arc_i for some i and thus $(v, v') \in \bigcup_{i \leq n} W_i(U_i \times V_i)$ holds.
- $v \leq v' \Rightarrow v \sqsubseteq_h v'$:
  If $v \leq v'$ then there is an $i \leq n$ and a word $w \in \mathsf{Arc}_i$ labeling the shortest path between them. Thus $w \in h(\sqsubseteq)$ and $v \sqsubseteq_h v'$ holds. □

As both representations describe the same set of uncolored linear orders, they can be used to represent the same set of colored linear colors if the colors themselves are finitely representable. This is clearly the case with regular colors and allows the following theorem:

Theorem 2
Let $I = (V, E, (P_c)_{c \in C})$ be a C-colored linear order with finitely many colors, then the following statements are equivalent:

- There is an interpretation $\mathscr{I}$ such that $\mathscr{I}(\mathfrak{T}_2) = I$
- There is an inverse substitution h, a regular language L and families of regular sets $Q, (P_c)_{c \in C}$ such that $\left(h^{-1}(\mathfrak{T}_2^Q) \restriction_L, (P_c)_{c \in C}\right) = I$
- I is prefix-recognizable.

3 Generating Representations

The representations of generators do not rely on extracting regularity from the complete binary tree. Instead they describe operations on trivially interpretable graphs to describe structure and use fixed points to impose infinite size.

3.1 Order Terms

Analogous to regular expressions we define expressions to generate colored linear orders.

Definition (Order terms)
Let C be a finite set of colors and $c \in C$. We define the set of *order terms* by the following grammar:

$$t ::= c \mid t + t \mid t \times \omega \mid t \times -\omega \mid \mathit{shuffle}(t, \ldots, t)$$

The + operator describes ordered sums, × ordered products and $shuffle(t_1, \ldots, t_n)$ the shuffle operation. c describes the linear order with one vertex that is colored solely with c. We define the order defined by a term with the evaluation function $[\![\ ]\!]$:

$$\begin{aligned}
[\![shuffle(t_1, \ldots, t_n)]\!] &= \sigma([\![t_1]\!], \ldots, [\![t_n]\!]) \\
[\![t \times -\omega]\!] &= [\![t]\!] \otimes -\omega \\
[\![t \times \omega]\!] &= [\![t]\!] \otimes \omega \\
[\![t + t']\!] &= [\![t]\!] \oplus [\![t']\!] \\
[\![c]\!] &= \big(\{0\}, \{(0,0)\}, (P_{c'})_{c' \in C}\big)
\end{aligned}$$

where $P_c = \{0\}$ and $P_{c'} = \emptyset$ for $c' \neq c$.

We show that all linear orders generated by order-terms can be described by interpretations in the binary tree.

Lemma 14
For every order term t there is an MSO-interpretation I such that $[\![t]\!] = I(\mathfrak{T}_2)$.

Proof. Induction on the structure of t:

- Base Case, $t = c$:
  In this case we can define the root element of the tree with $\delta(x) := \forall z.\big(x \neq left(z) \wedge x \neq right(z)\big)$ and define the relations by $\phi_{\leq}(x, y) = \phi_c(x) = true$ and $\phi_{c'} := false$ if $c' \neq c$. Finally we set $I = \langle \delta, \phi_{\leq}, (\phi_c)_{c \in C} \rangle$.

- Step Case, $t = t' + t''$:
  By induction hypothesis we have interpretations I', I'' for $[\![t']\!]$,$[\![t'']\!]$ with $I' = \langle \delta', \phi'_{\leq}, (\phi'_c)_{c \in C} \rangle$ and $I'' = \langle \delta'', \phi''_{\leq}, (\phi''_c)_{c \in C} \rangle$.

  We need to ensure that we do not get elements satisfying δ that are in both orders, thus we evaluate δ' on the left sub-tree of the root and δ'' on the right one:

  $$\delta(x) = \delta'^{\left(left(\epsilon)\right)}(x) \vee \delta''^{\left(right(\epsilon)\right)}(x)$$

  This way we get $[\![t']\!]$ as an order of elements in the left sub-tree and $[\![t'']\!]$ an order of elements in the right sub-tree, and we can distinguish between them by checking whether 0 or 1 is a prefix of all their elements.

  As we have shifted the elements one step down we also need to consider this in the coloring:

  $$\phi_c(x) = \phi_c'^{\left(left(\epsilon)\right)}(x) \vee \phi_c''^{\left(right(\epsilon)\right)}(x)$$

  However we can just use the ordering from the tree to encode the order on the interpreted order:

  $$\phi_{\leq}(x, y) := \big(left(\epsilon) \leq x \wedge right(\epsilon) \leq y\big) \vee \phi'^{\left(left(\epsilon)\right)}_{\leq}(x, y) \vee \phi''^{\left(right(\epsilon)\right)}_{\leq}(x, y)$$

- Step Case, $t = t' \times \omega$:
  By induction hypothesis we have an interpretation I' for $[\![t']\!]$ with $I' = \langle \delta', \phi'_{\leq}, (\phi'_c)_{c \in C} \rangle$.

  One can see $t' \times \omega$ as a shortcut for $t' + t' + t' + \ldots$. We can generalize the encoding for addition by choosing not two elements where we interpret the sub-tree but by choosing a infinite descending chain and interpreting the tree of t' under each element of it.

  $$\delta(x) := \exists y. \left[\mathit{RBranch}(y) \wedge \delta'^{\left(\mathit{left}(y)\right)}(x) \right]$$

  This formula expresses that there is a vertex on the rightmost path of the tree, and we evaluate δ' in the left subtree of any of this vertex.

  The predicates for the relations are analogous to the $t + t$ case.

  $$\begin{aligned} \phi_c(x) &:= \forall y. \left[\mathit{RBranch}(y) \wedge \phi_c'^{\left(\mathit{left}(y)\right)}(x) \right] \\ \phi_{\leq}(x,y) &:= \exists z.\ \exists z'. \Big[\mathit{RBranch}(z) \wedge \mathit{RBranch}(z') \wedge \mathit{left}(z) \leq x \wedge \mathit{left}(z') \leq y \wedge \\ &\qquad \left((z = z' \wedge \phi'^{\left(\mathit{left}(z)\right)}_{\leq}(x,y)) \vee (z \neq z' \wedge z \leq z') \right) \Big] \end{aligned}$$

- Step Case, $t = t' \times -\omega$:
  analogous to the case $t' \times \omega$ with exchanging *left* and *RBranch* with *right* and *LBranch*.

- Step Case, $t = \mathit{shuffle}(t^{(1)}, \ldots, t^{(n)})$:
  By induction hypothesis we have for each $i \leq n$ interpretations $I^{(i)}$ for $[\![t^{(i)}]\!]$ with $I^{(i)} = \langle \delta_{(i)}, \phi_{\leq (i)}, (\phi_{c(i)})_{c \in C} \rangle$.

  By Lemma 3 we can construct n dense pairwise disjoint MSO-definable sets in $\mathfrak{T}_2$ such that every right successor of every element in every of these maps contains no elements from any of these sets. We use the formulas $(\psi_i)_{i \leq n}$ given in the proof of the lemma and interpret the right successor of every element in the ith set with a copy of $[\![t^{(i)}]\!]$:

  $$\delta(x) := \exists y. \bigvee_{i \leq n} \left(\psi_i(y) \rightarrow \delta_{(i)}^{\left(\mathit{right}(y)\right)}(x) \right)$$

  The predicate formulas are constructed analogously:

  $$\begin{aligned} \phi_c(x) &:= \exists y. \Big[\bigvee_{i \leq n} \left(\psi_i(y) \rightarrow \phi_{c_{(i)}}^{\left(\mathit{right}(y)\right)}(x) \right) \Big] \\ \phi_{\leq}(x,y) &:= \exists z. \exists z'. \Big[\bigvee_{i \leq n} \psi_i(z) \wedge \bigvee_{i \leq n} \psi_i(z') \wedge \mathit{right}(z) \leq x \wedge \mathit{right}(z') \leq x \wedge \\ &\qquad \left((z = z' \wedge \phi'_{\leq}(x,y)) \vee (z \neq z' \wedge z \leq z') \right) \Big] \end{aligned}$$

□

3.2 Systems of Graph Equations

Order terms as introduced in the last section reassemble regular expressions for a more general class of infinite linear orders. Here we introduce an extension of *grammars* to this class.

Definition
Let $X = \{x_0, x_1, \ldots, x_n\}$ be a finite set of non-terminal symbols and C a set of colors. The set of *equation terms* t is defined by the following grammar:

$$t ::= c \mid x_i \mid t + t$$

An *order equation* is an expression $x_j = t$ where t is an equation term.

An *system of order equations* is a tuple (X, C, S, x_0) where X and C are as above, S is a set of order equations and $x_0 \in X$ is the *start variable*. Every non-terminal symbol occurs exactly once on the left-hand-side of an equation.

We call a system of equations *simple* iff every equation has the form $x_i = x_j + x_k$ or $x_i = c$.

We assume fixed X, C in this section and write S in the usual form:

$$\begin{aligned} x_0 &= x_j + x_k \\ &\cdots \\ x_i &= c_i \\ &\cdots \\ x_n &= x_l + x_m \end{aligned}$$

In regular grammars the solution of such a system is obtained by replacing non-terminals with their definition. Here the non-terminals are *nodes* colored with an element of X that have to be replaced with the order described by this symbol.

Definition
Let $G = (X, C, S, x_0)$ be a system of equations. The *starting configuration* of G is a set of orders $(g_x)_{x \in X}$ with colors $C \cup X$ where every g_x is

- the graph with a single node that is colored with c if x occurs on the left-hand side of an equation in S as $x = c$
- the graph with two nodes, colored with x_i or x_k if x occurs on the left-hand side of an equation in S as $x = x_i + x_k$. In this graph there is a single edge going from the node colored with x_i to the other one.

Let $\hat{G} = (g_x)_{x \in X}$ be such a system of orders with colors $C \cup X$. The *step-function* $\mathscr{S}_{\hat{G}}$ takes one order as input and replaces all of its non-terminal nodes, which are colored with x_i by the order g_{x_i}. If there is no non-terminal node it returns the order unchanged. The *translation function* $\mathscr{S}^*$ takes a system of orders as described above and applies the step-function to every order.

We aim to define the solution of a system of equations G as sub-graph of the least fixed point of $\mathscr{S}^*$ that is obtained from the starting variable. Before we can do this we need to state some properties of $\mathscr{S}_{\hat{G}}$:

Lemma 15
If we regard the set of all linear orders as a partial order with the prefix ordering the following two statements hold:

- $\mathscr{S}_{\hat{G}}$ is monotonous
- $\mathscr{S}_{\hat{G}}$ is ω-complete, i.e. every sequence has a least upper bound.

A proof can be found in [CE12]

Definition
Let G be a system of equations and $(g_x)_{x\in X}$ be a fixed point of $\mathscr{S}^*$, then g_{x_0} is a *solution* of G. The canonical solution is the graph g'_{x_0} of the least fixed point $(g'_x)_{x\in X}$ of $\mathscr{S}^*$. We denote the canonical solution by $\mathsf{can}(G)$.

By Lemma 15 we can conclude that there is always a canonical solution. We show now that the canonical solution can be described with an order term:

Lemma 16
Given a simple system of equations $G = (X, C, S, x_0)$ we can construct an order term t such that $[\![t]\!] = \mathsf{can}(G)$

Proof. We define on the set X the following partial order: If $x_k = x_i + x_j$ then $x_k \sqsubseteq x_i$ and $x_k \sqsubseteq x_j$. The undirected graph of this order may not be connected, but for the construction of the term only the sub-graph which contains x_0 is relevant. Thus we may assume that $\sqsubseteq$ forms a connected graph where all nodes are reachable from the node x_0.

If this graph is a tree with x_0 as the root, we may regard it as the syntax tree representation of a term: the leaves are non-terminals with equations $x_i = c$ and we identify their node with the respective color. The inner nodes are non-terminals with equations $x_k = x_i + x_j$ which we identify with $+$.

If this graph is not a tree but contains no cycles we turn it into one: Let x be a node that has two predecessors. We copy the graph under x and redirect one of the edges (x', x) to this copy.

Hence we only must find terms that resolve circles in the $\sqsubseteq$-graph and turns it into a tree. In any case we first remove the chains ending in leaves: we repeatedly replace all x_k whose equation contains no non-terminal by their defining equation-term. Thus we may assume that the circle has no outgoing edges.

- Case 1 - the circle is a loop.
  In this case the node with the loop is a variable x_k with an equation $x_k = x_k + c$ or $x_k = c + x_k$. We remove the loop and mark the node with the solution of its equation: $c \times \omega$ resp. $c \times (-\omega)$
- Case 2 - all circles are disjoint.
  I.e. every non-terminal occurs in at most one circle. In this case the non-terminals

$x_i, x_j, \ldots x_k$ form a subsystem of equations where every equation has one of the following forms:

$$x_i = x_j + y$$
$$x_j = y' + x_l$$

where y, y' are equation terms without non-terminals. We repeatedly replace all x_k in this sub system by its definition until we have a single equation of the form

$$x_k = y + x_k + y'$$

where y, y' are equation terms without non-terminals as before. We remove the whole loop and replace it with a single node x_k which we mark with the solution of this equation: $y \times \omega + y' \times (-\omega)$.

Case 1 can be seen as a degenerate subcase of this.

- Case 3 - some circles intersect.
  I.e. there are nodes that occur in two circles. In this case the non-terminals $x_i, x_j, \ldots x_k$ of the both circles form an sub system where every equation has one of the following forms:

$$x_i = x_k + y$$
$$x_j = y + x_l$$
$$x_k = x_m + x_n$$

We first apply the method from case 2 to get rid of the variables that have equation of the first two forms. We end up with a system of equations where every equation has the following form:

$$x_k = y + x_m + y' + x_n + y''$$

where y, y', y'' are equation terms without non-terminals as before and x_n, x_m have equations of the same form.

For each of these equations we introduce a new variable x'_k and split the equation:

$$x_k = y + x_m + x'_k$$
$$x'_k = y' + x_n + y''$$

And replace x_k in the new equation with its definition, thus we end up with a system of equations of the following form:

$$x_k = y + x_m + x_n + y'$$

For each of these equations we introduce new variables λ_k, ρ_k, μ_k and redefine the equation:

$$\lambda_k = y + \lambda_m$$
$$\rho_k = \rho_n + y'$$
$$\mu_k = \mu_m + \rho_m + \lambda_n + \mu_n$$
$$x_k = \lambda_k + \mu_k + \rho_k$$

The solution of the first two equations can be obtained from case 2:

$$\lambda_k = (y_0 + \cdots + y_i) \times \omega$$
$$\rho_k = (y'_0 + \cdots + y'_i) \times (-\omega)$$

Where $y_0, y'_0 \ldots$ are all the y and y' of the above equations. The μ_k are the dense shuffle of the $\rho_m + \lambda_n$, thus we can replace all the cycles involved in the construction by n new nodes x^*_k, one for each in-going edge. Each x^*_k is marked with the following term:

$$(y_0 + \cdots + y_i) \times \omega + \mathit{shuffle}(\rho_l \lambda_k) + (y'_0 + \cdots + y'_i) \times (-\omega)$$

Where $\rho_l \lambda_k$ is the shortcut for the list of all $\rho_m + \lambda_n$ with n, m for all pairs (x_n, x_m) of variables that had an equation in the last step of the construction.

After removing all cycles we can again regard the tree as a syntax tree of a term, but with the aforementioned markings as the terms on the leaves. □

We complete this section by showing that given an interpretation for a linear order we can construct a system of order equations that has this linear order as its solution. First we define syntactic congruences for our orders:

Definition
As seen in Theorem 2 each linear order interpretable in $\mathfrak{T}_2$ is composed from regular languages. The *syntactic congruence* $\sim$ of such a linear order $\mathfrak{A}$ with respect to a fixed interpretation is the intersection of all syntactical congruences of the languages of its representation as described in Lemma 13.

Lemma 17
Given an interpretation $I = \langle \delta(x), \phi_{\leq}(x, y), (\phi_c(x))_{c \in C} \rangle$ we can construct a simple system of order equations $G = (X, C, S, x_0)$ such that

$$I(\mathfrak{T}_2) = \mathsf{can}(G)$$

Proof. Given Lemma 5 we can assume that $I(\mathfrak{T}_2) = (V, \leq, (P_c)_{c \in C})$ where $\leq$ is a suborder of $\leq_{\mathfrak{T}_2}$. We first construct an infinite system of order equations by assigning a non-terminal A_v to each vertex $v \in \{0,1\}^*$ with:

$$A_v = \begin{cases} A_{v0} + c + A_{v1} & \text{if } v \in V \wedge v \in P_c \\ A_{v0} + A_{v1} & \text{otherwise} \end{cases}$$

It is easy to see that this system characterizes $I(\mathfrak{T}_2)$. We can reduce this system to a finite one because there are only finitely many classes of pairs of disjoint subtrees in $\mathfrak{T}_2$ which are equal up to isomorphism [Blu03]: This follows from the fact that $t_1 \sim t_2$ implies $t_1 \cong t_2$ and there are only finitely many $\sim$ classes [Hol82] and thus $\diagdown v \sim \diagdown w \wedge v \sim w$ implies that the tree under v and w are isomorphic.

Thus we can group the equations by the equivalence class of their left-hand side and retain only one equation for each equivalence class. Finally we turn equations of the form $A = A' + c + A''$ into a subsystem of the form

$$\begin{aligned} A &= A' + B \\ B &= C + A'' \\ C &= c \end{aligned}$$

with new non-terminals B, C. Thus the resulting system has $I(\mathfrak{T}_2)$ as its solution and is finite and simple. □

Given this three lemmas we can state the following theorem:

Theorem 3
Let L be a C-colored linear order with finitely many colors, then the following statements are equivalent:

- There is an interpretation I such that $I(\mathfrak{T}_2) = L$
- There is a simple system of order equations G such that $\mathsf{can}(G) = L$
- There is an order term t such that $[\![t]\!] = L$

4 Generalizations and Further Representations

The representations of acceptors in this work are based on the study of general graphs interpretable in the complete binary tree. Thus Theorem 2 holds for this bigger class. Our proof of 2 is based on a general proof in [Cau96]. This class of graphs can be extended in several ways: [Cau02] introduced a hierarchy with the prefix-recognizable graphs at the bottom. In [KN95] automatic structures were introduced, which characterize infinite structures that are FO-interpretable in the complete binary tree with additional information. The prefix-recognizable graphs are a strict subset of them.

The representations of acceptors require another set of operations to generalize to general graphs. In [Bar97] it is shown that there is such a set that gives rise to systems of equations whose solutions are prefix-recognizable. These VR-grammars generalize the simple graph grammars used here. Further graph terms are introduced, where the countable terms have prefix-recognizable graphs as their solution, whether it is possible to represent them with finite terms over other operations is yet unknown. Our proof is based on [Blu03].

In [Blu01] representations based on groups for prefix-recognizable graphs are presented, e.g. that the Caylay-graph of a group is prefix-recognizable iff the group is context-free or virtually free.

Furthermore it is known that prefix-recognizable graphs correspond to the configuration graph of pushdown automata with ε-transitions[Sti00].

References

[Bar97] Klaus Barthelmann. On equational simple graphs. Technical report, 1997.

[Blu01] Achim Blumensath. Prefix-recognisable graphs and monadic second-order logic. Technical report, RWTH Aachen, 2001.

[Blu03] Achim Blumensath. *Structures of Bounded Partition Width*. PhD thesis, RWTH Aachen, 2003.

[Cau96] Didier Caucal. On infinite transition graphs having a decidable monadic theory. In *Proceedings of the 23rd International Colloquium on Automata, Languages and Programming*, ICALP '96, pages 194–205, London, UK, UK, 1996. Springer-Verlag.

[Cau02] Didier Caucal. On infinite terms having a decidable monadic theory. In *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*, pages 165–176, 2002.

[CE12] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*. Cambridge University Press, New York, NY, USA, 1st edition, 2012.

[EB58] Calvin C. Elgot and Julius R. Büchi. Decision problems of weak second order arithmetics and finite automata. *Notices of the American Mathematical Society*, 1958.

[ER66] Calvin C. Elgot and Michael O. Rabin. Decidability and undecidability of extensions of second (first) order theory of (generalized) successor. *Journal of Symbolic Logic*, 31(2):169–181, 1966.

[GKP94] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 1994.

[Hol82] William Michael Lloyd Holcombe. *Algebraic automata theory*. Cambridge studies in advanced mathematics. Cambridge University Press, Cambridge, New York, Melbourne, 1982.

[KN95] Bakhadyr Khoussainov and Anil Nerode. Automatic presentations of structures. In *Selected Papers from the International Workshop on Logical and Computational Complexity*, LCC '94, pages 367–392, London, UK, UK, 1995. Springer-Verlag.

[Ros82] J.G. Rosenstein. *Linear Orderings*. 1982.

[Sti00] Colin Stirling. Decidability of bisimulation equivalence for pushdown processes. Technical report, 2000.

[Tho96] Wolfgang Thomas. Languages, automata, and logic. In *Handbook of Formal Languages*, pages 389–455. Springer, 1996.