Decidability of MSO Uniformisation on Finite Trees

Thomas Colcombet
CNRS, IRIF, Université Paris Diderot, France

Yago Iglesias Vázquez
Université Paris Diderot, France

Abstract

We study uniformisation for MSO formulas over unordered finite binary trees. A formula $\psi(X)$ uniformises $\varphi(X)$ if it selects at most one witness X and, whenever $\varphi(X)$ has a witness, $\psi(X)$ picks one. We show that, using symmetry-preserving automata (order-insensitive automata), it is decidable whether a given $\varphi(X)$ admits such a uniformiser, and in the positive case, we effectively construct $\psi(X)$. Our procedure constructs from $\varphi(X)$ an order-insensitive deterministic bottom-up tree automaton, checks whether that automaton accepts every tree, and, when it does, extracts from its transition relation an MSO-formula that uniformly selects a witness. This reduces MSO uniformisation on finite trees to a simple automata universality test and a formula extraction step.

2012 ACM Subject Classification Theory of computation → Finite Model Theory; Theory of computation → Algebraic language theory; Theory of computation → Regular languages

Keywords and phrases MSO, uniformisation, finite state automata, trees

Digital Object Identifier 10.4230/LIPIcs...

1 Introduction

Monadic second-order logic (MSO for short) is a well-studied fragment of second-order logic with wide-ranging applications in graph theory, automata theory, and formal verification. In particular, its deep connection with automata theory has been extensively explored. It is well known that MSO over finite words is equivalent in expressive power to finite-state automata [1].

This correspondence extends beyond words: automata can also be defined to run on trees. Over finite trees, MSO logic remains equivalent in expressive power to finite-state tree automata [9, 5]. This connection enables the use of automata-theoretic techniques to reason about trees, one of the most fundamental and widely studied data structures.

A classical challenge in this context is the uniformisation problem: given an MSO-formula $\varphi(X)$ that asserts the existence of a set X satisfying a certain property, is it possible to define another MSO-formula $\psi(Y)$ that selects *at most one* such set for each tree, in a way that agrees with $\varphi(X)$ whenever it is satisfiable? We formalize this notion in Section 3, Definition 44.

This problem has been studied in various settings. A comprehensive survey is provided in [2]. Positive results are known for finite words [3], and infinite words, where every MSO-definable relation admits an MSO-definable uniformising function [10, 3, 8]. In contrast, uniformisation is not possible in general for infinite binary trees [6]. In this work, we study MSO uniformisation over a different class of structures: *finite, unordered, labeled binary trees.* All of the technical work is performed using only automata that preserve structural symmetry, *i.e.* they are unable to distinguish between left and right subtrees.

The central question addressed in this paper is:

▶ **Problem 1** (uniformisation problem). *Given an MSO-formula $\varphi(X)$, does there exist a uniformiser $\psi(X)$?*

Our main result, Theorem 46, is that this question is *decidable*, and in the positive case, a uniformiser $\psi(X)$ can be effectively constructed. Concretely, we translate $\varphi(X)$ into a deterministic bottom-up tree automaton whose transition map is order-insensitive – closed under sibling swaps. We then perform a universality check on this automaton – does it accept every tree? – and, when it does, we extract from its transitions an MSO-formula that uniformly selects a witness.

This paper makes two main contributions:

- It demonstrates the equivalence between MSO uniformisation, definability, and automorphism invariance, Lemma 47,
- It extends the decidability of MSO uniformisation to finite binary trees, Theorem 46.

Structure of the paper.

The remainder of the paper is structured as follows: Section 2 introduces the necessary definitions, notations and classical results on trees, automata, and MSO logic, with an emphasis on the order-insensitive variant of tree automata. In Section 3, we define uniformisation and present our main result, proving that the uniformisation problem is decidable for unordered finite binary trees.

In this paper notions are hyperlinked to their definitions, reading the paper in electronic form is recommended.

2 Definitions

In this section, we review classical concepts related to trees, automata, and logic. Section 2.1 establishes our notation for trees. Section 2.2 provides a brief overview of bottom-up tree automata. Section 2.3 introduces an order-insensitive variant of these automata, and Section 2.4 defines a corresponding universality test. Finally, Section 2.5 describes how to translate between MSO formulas and order-insensitive automata.

2.1 Trees

This first subsection contains the definition of the trees we are working with. Other equivalent definitions of trees and automata are possible [11, 4], but we have chosen this one because it is more intuitive, without adding much complexity to the proofs.

▶ **Definition 1.** *A tree over an alphabet Σ is recursively defined as follows:*

- *a, where $a \in \Sigma$;*
- *$a(t, t')$, where $a \in \Sigma$ and t, t' are trees.*

We call the root of a tree the first node in its definition. We denote the children of a tree t as t' and t'' in the case of $t = a(t', t'')$. The set of all trees over Σ is denoted Tree_Σ. A subset of Tree_Σ is called a tree language (or language for short).

This recursive structure induces a natural notion of position within a tree.

▶ **Definition 2.** *The set of nodes of a tree t, denoted $\mathrm{Nodes}\,(t)$, is defined as the set of binary strings (words over $\{0, 1\}$) that represent paths from the root to each node:*

$$\mathrm{Nodes}\,(t) = \begin{cases} \{\varepsilon\} & \textit{if } t = a \\ \{\varepsilon\} \cup \{0u \mid u \in \mathrm{Nodes}\,(t')\} \cup \{1u \mid u \in \mathrm{Nodes}\,(t'')\} & \textit{if } t = a(t', t'') \end{cases}$$

A node is thus the path from the root to a particular position in the tree.

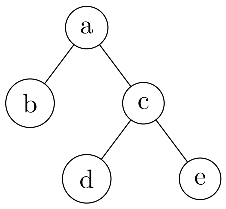


Figure 1 Concrete tree corresponding to $a(b, c(d, e))$.

▶ **Example 3.** Let $t := a(b, c(d, e))$. We compute:

$$\text{Nodes}(b) = \{\varepsilon\}, \quad \text{Nodes}(c(d,e)) = \{\varepsilon, 0, 1\}$$

$$\text{Nodes}(t) = \{\varepsilon\} \cup \{0\} \cup \{1, 10, 11\} = \{\varepsilon, 0, 1, 10, 11\}$$

Among the nodes, some are distinguished as leaves, nodes without children.

▶ **Definition 4.** *The leaves of a tree t are defined as follows:*

$$\text{Leaves}(t) = \begin{cases} \varepsilon & \textit{if } t = a \\ \{0u \mid u \in \text{Leaves}(t')\} \cup \{1u \mid u \in \text{Leaves}(t'')\} & \textit{if } t = a(t', t'') \end{cases}$$

We can see that $\text{Leaves}(t) \subseteq \text{Nodes}(t)$ *and we call internal nodes the set* $\text{Nodes}(t) \cap \text{Leaves}(t)$

Each node in a tree carries a symbol from the alphabet. Given a tree t , we can access the *label* of a node thanks to the function $t(\cdot)$:

$$t(n) = \begin{cases} a & \text{if } t = a \text{ and } n = \varepsilon \\ a & \text{if } t = a(t', t'') \text{ and } n = \varepsilon \\ t'(m) & \text{if } t = a(t', t'') \text{ and } n = 0m \\ t''(m) & \text{if } t = a(t', t'') \text{ and } n = 1m \end{cases}$$

We often need to refer to the subtree rooted at a particular node. We introduce the notation t^n to refer to the *subtree* of t rooted at the node n. This can be computed as follows:

$$t^n = \begin{cases} t & \text{if } n = \varepsilon \\ t'^m & \text{if } t = a(t', t'') \text{ and } n = 0m \\ t''^m & \text{if } t = a(t', t'') \text{ and } n = 1m \end{cases}$$

The ancestor relation captures the natural hierarchical structure of a tree: a node x is an ancestor of a node y if y is located in a subtree rooted at x.

▶ **Definition 5.** *Let t be a tree and $x, y \in \text{Nodes}(t)$. We define the ancestor relation as:*

$x \sqsubseteq y$ *if and only if* $\exists z \in \text{Nodes}(t), y = xz$

▶ Remark 6. Note that this order does not depend on the left or right position of the children, *i.e.* it preserves order-insensitivity, this will be explored in the following sections.

2.2 Automata over trees

In this subsection we aim to define the basics of tree automata in a comprehensive way, adding as much detail as possible to the proofs. For reference, one can look at more advanced books on this topic [4, 7] and the survey [11].

In classical automata theory, we analyze deterministic finite automaton (DFA), wich assign a unique state to each prefix of a word, updating this state at each step using a transition function, ussualy reading the input from left to right.

A deterministic bottom-up tree automaton (DBUA) generalizes this idea to trees: instead of reading a linear word, it reads a a tree where each node carries a symbol from a fixed alphabet. Rather than scanning from left to right, it proceeds bottom-up, computing states at each node by combining the states of its children : the automaton aggregates the information from those subtrees to assign a state to the parent.

The idea is to recursively assign a state to each node of the tree:

- At the leaves, the state is determined directly from the symbol via a function ι, just as a DFA starts in a particular initial state.
- A DFA transitions based on the current letter and the state reached by the previous letters. Similarly, when a DBUA reaches an internal node, it transitions based on the states reached by its children and the label associated to the node.
- Finally, the state assigned to the root tells us whether the whole tree is accepted, depending on whether it belongs to a set of final states.

We now state the formal definitions.

▶ **Definition 7.** *A deterministic bottom-up tree automaton (DBUA) is a tuple* $(\Sigma, Q, \iota, \delta, F)$ *where:*

- Σ *is an alphabet.*
- Q *is a finite set of states.*
- $\iota : \Sigma \to Q$ *is a function that initializes the initial states of the leaves.*
- $\delta : Q \times Q \times \Sigma \to Q$ *is the transition map.*
- $F \subseteq Q$ *is the set of final states.*

We often use ι_a *to denote* $\iota(a)$ *for* $a \in \Sigma$ *and* δ_a *to denote* $\delta(\cdot, \cdot, a)$ *for* $a \in \Sigma$.

▶ **Definition 8.** *The interpretation of a DBUA* $A = (\Sigma, Q, \iota, \delta, F)$ *over an alphabet* Σ *is defined as follows:*

$$\begin{array}{rcl} [\![A]\!] : \mathrm{Tree}_\Sigma & \to & Q \\ a & \mapsto & \iota_a \\ a(t, t') & \mapsto & \delta_a([\![A]\!](t), [\![A]\!](t')) \end{array}$$

We say that A *accepts a tree* t *if* $[\![A]\!](t) \in F$.
We will sometimes use the notation $\delta(t)$ *for the interpretation of* A *over the tree* t.

▶ **Example 9** (trees with at most two leaves)**.** We define a DBUA $A = (\Sigma, Q, \iota, \delta, F)$ that accepts exactly the trees with at most two leaves. Let:

- $Q = \{0, 1, 2\}$, where $q = 1$ means *one leaf*, $q = 2$ means *two leaves*, and $q = 3$ means *more than two leaf*.
- $F = \{1, 2\}$.
- For all $a \in \Sigma$, we define $\iota_a = 1$.
- The transition function is defined by $\delta_a(q_1, q_2) = \min(q_1 + q_2, 3)$.

Intuitively, this automaton counts the number of leaves in the tree, but caps the count at 3 to detect when the number exceeds 2. Figure 2 illustrates the execution of this automaton over a particular input.

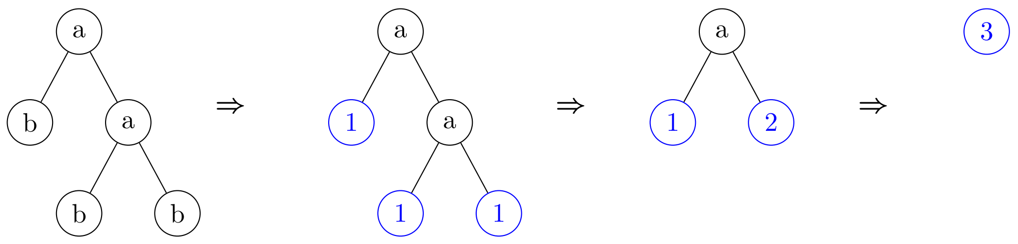


Figure 2 Execution of the automaton of Example 9 over the input $a(b, a(b, b))$

In the same way that we can define a non-deterministic automaton over strings, the same can be done for tree automata using the same techniques. We will define them and give basic definitions on how to use them.

▶ **Definition 10.** *A non-deterministic bottom-up tree automaton (NBUA) is a tuple* $(\Sigma, Q, \mathcal{I}, \Delta, F)$ *where:*

- Σ *is an alphabet.*
- Q *is a finite set of states.*
- $\mathcal{I} \subseteq \Sigma \times Q$ *correspond to the possible states of the leaves.*
- $\Delta \subseteq Q \times Q \times \Sigma \times Q$ *is the transition relation.*
- $F \subseteq Q$ *is the set of final states.*

▶ **Definition 11.** *A run* ρ *of an NBUA* $A = (\Sigma, Q, \mathcal{I}, \Delta, F)$ *over a tree* t *is :*

$$\begin{array}{rcl} \rho : \mathrm{Nodes}\,(t) & \to & Q \\ (t\,(b)\,, \rho(n)) \in \mathcal{I} & \textit{if} & n \in \mathrm{Leaves}\,(t) \\ (\rho(n0), \rho(n1), t\,(n)\,, \rho(n)) \in \Delta & \textit{if} & n \in \mathrm{Nodes}\,(t) \setminus \mathrm{Leaves}\,(t) \end{array}$$

We say that ρ *is an accepting run if* $\rho(\varepsilon) \in F$ *and* A *accepts* t *if* $F \cap \{\rho(\varepsilon) \mid \rho \text{ run of } A \text{ over } t\} \neq \emptyset$

▶ **Definition 12.** *Let* A *be an NBUA. Its associated language is defined as:*

$$\mathcal{L}\,(A) = \{t \in \mathrm{Tree}_\Sigma \mid \textit{exists } \rho \textit{ a accepting run of } A \textit{ over } t\}$$

▶ Remark 13. Just like in classical automata theory, we can see a DBUA $A = (\Sigma, Q, \iota, \delta, F)$ as a special case of a NBUA $A' = (\Sigma, Q, \mathcal{I}, \Delta, F)$ where:

- $\mathcal{I} = \{(a, \iota_a) \mid a \in \Sigma\}$,
- $\Delta = \{(q_1, q_2, a, \delta_a(q_1, q_2)) \mid q_1, q_2 \in Q, a \in \Sigma\}$.

In this case, A' has exactly one possible run on each tree :

$$\begin{array}{rcl} \rho : \mathrm{Nodes}\,(t) & \to & Q \\ n & \mapsto & \iota_{t(n)} \text{ if } n \in \mathrm{Leaves}\,(t) \\ n & \mapsto & \delta_{t(n)}(\rho(n0), \rho(n1)) \text{ if } n \in \mathrm{Nodes}\,(t) \setminus \mathrm{Leaves}\,(t) \end{array}$$

The following lemma shows the equivalence of both definitions.

▶ **Lemma 14.** *Let A be a DBUA,*

$$\mathcal{L}(A) = \{t \in \mathrm{Tree}_\Sigma \mid [\![A]\!](t) \in F\}$$

Proof. We will prove by induction on a tree t that (a) there exists a run of A over t, and (b) for all runs ρ of A over t, $[\![A]\!](t) = \rho(\varepsilon)$. We proceed by case distinction.

- If $t = a$. Let ρ be defined by $\rho(\varepsilon) = \iota_a$, then ρ is a run of A over t. We have proved (a). Consider now some run ρ of A over t. We have $\rho(\varepsilon) = \iota_a = [\![A]\!](a) = [\![A]\!](t)$. We have proved (b).
- If $t = a(t', t'')$. By induction hypothesis (a), there exists runs ρ' and ρ'' on t' and t'' respectively. Let ρ be defined by $\rho(\varepsilon) = \delta_{t(\varepsilon)}(\rho'(\varepsilon), \rho''(\varepsilon))$, $\rho(0u) = \rho'(u)$ for all $u \in \mathrm{Nodes}(t')$ and $\rho(1u) = \rho''(u)$ for all $u \in \mathrm{Nodes}(t'')$. Then ρ is a run of A over t. We have proved (a).
  Consider now some run ρ of A over t. Let $\rho'(u) = \rho(0u)$ for all $u \in \mathrm{Nodes}(t')$ and $\rho''(u) = \rho(0u)$ for all $u \in \mathrm{Nodes}(t'')$. Then ρ' is a run of A over t' and ρ'' over t''. By induction hypothesis (b) twice, we know that $\rho'(\varepsilon) = [\![A]\!](t')$ and $\rho''(\varepsilon) = [\![A]\!](t'')$. We have now $\rho(\varepsilon) = \delta_{t(\varepsilon)}(\rho'(\varepsilon), \rho''(\varepsilon)) = \delta_{t(\varepsilon)}([\![A]\!](t'), [\![A]\!](t'')) = [\![A]\!](t)$. We have proved (b).

◀

We aim now to show that, like in the classical theory, both non-deterministic automata and deterministic ones have the same expressiveness power. Since DBUA is a particular case of NBUA it suffices to show that each language accepted by an NBUA can also be accepted by a DBUA.

▶ **Definition 15.** *Let $A = (\Sigma, Q, I, \Delta, F)$ be an NBUA, we define the* power set automaton *of A, noted* Det(A)*, as* $\mathrm{Det}(A) = (\Sigma, \mathcal{P}(Q), \iota, \delta, F')$ *where:*

- $\iota : \Sigma \to \mathcal{P}(Q)$
  $$a \mapsto \{q \in Q \mid (a, q) \in \mathcal{I}\}$$
- $\delta : \mathcal{P}(Q) \times \mathcal{P}(Q) \times \Sigma \to \mathcal{P}(Q)$
  $$(X, Y, a) \mapsto \{q \in Q \mid (x, y, a, q) \in \Delta, x \in X, y \in Y\}$$
- $F' = \{X \in \mathcal{P}(Q) \mid X \cap F \neq \emptyset\}$

▶ **Theorem 16.** *For all $N = (\Sigma, Q, I, \Delta, F)$ an NBUA, the power set automaton verifies that*

$$[\![\mathrm{Det}(N)]\!](t) = \{\rho(\varepsilon) \mid \rho \textit{ run of } N \textit{ over } t\} \quad \textit{and } \mathcal{L}(N) = \mathcal{L}(\mathrm{Det}(N))$$

Proof. We will first show that for all trees t,

$$[\![\mathrm{Det}(N)]\!](t) = \{\rho(\varepsilon) \mid \rho \text{ run of } N \text{ over } t\} \ .$$

We will prove this property by induction over t. We proceed by case distinction.

- If $t = a$, then
  $$\begin{aligned} \{\rho(\varepsilon) \mid \rho \text{ run of } N \text{ over } t\} &= \{q \in Q \mid (a, q) \in \mathcal{I}\} \\ &= [\![\mathrm{Det}(N)]\!](t) \ . \end{aligned}$$
- If $t = a(t_1, t_2)$, then by the induction hypothesis we know that $[\![\mathrm{Det}(N)]\!](t_i) = q_i$ if and only if there exists a run over t_i ending at q_i, for $i \in \{1, 2\}$.

$$\begin{array}{ll}
 & q \in [\![\mathrm{Det}\,(N)]\!]\,(t) \\
\text{if and only if} & q \in \delta_a([\![\mathrm{Det}\,(N)]\!]\,(t_1)\,, [\![\mathrm{Det}\,(N)]\!]\,(t_2)) \\
\text{if and only if} & q \in \{q \in Q \mid (x_1, x_2, a, q) \in \Delta, x_1 \in [\![\mathrm{Det}\,(N)]\!]\,(t_1)\,, x_2 \in [\![\mathrm{Det}\,(N)]\!]\,(t_2)\} \\
\text{if and only if} & \exists x_1, x_2, (x_1, x_2, a, q) \in \Delta, x_1 \in [\![\mathrm{Det}\,(N)]\!]\,(t_1)\,, x_2 \in [\![\mathrm{Det}\,(N)]\!]\,(t_2) \\
\text{if and only if} & \text{there exists a run over } t_i \text{ ending at } x_i \text{ for } i \in \{1,2\} \text{ and } (x_1, x_2, a, q) \in \Delta \\
\text{if and only if} & q \in \{\rho(\varepsilon) \mid \rho \text{ run of } N \text{ over } t\}
\end{array}$$

Then $[\![\mathrm{Det}\,(N)]\!]\,(t) = \{\rho(\varepsilon) \mid \rho \text{ run of } N \text{ over } t\}$.

We must now show that $\mathrm{Det}\,(N)$ accepts t if and only if N accepts t.

$$\begin{array}{lll}
N \text{ accepts } t & \text{if and only if} & F \cap \{q \in Q \mid (a, q) \in \mathcal{I}\} \neq \emptyset \\
 & \text{if and only if} & F \cap [\![\mathrm{Det}\,(N)]\!]\,(t) \neq \emptyset \\
 & \text{if and only if} & [\![\mathrm{Det}\,(N)]\!]\,(t) \in F' \\
 & \text{if and only if} & \mathrm{Det}\,(N) \text{ accepts } t
\end{array}$$

and so $\mathcal{L}\,(N) = \mathcal{L}\,(\mathrm{Det}\,(N))$. ◀

2.3 Order-insensitivity

We will define a particular kind of tree automata: the ones that do not distinguish between left and right children. That means that the state reached from the two children is the same if we swap them. These automata are interesting because they respect the symmetries of trees, which will pay an important role in Section 3. While our definition of trees is ordered, *i.e.* we can syntactically distinguish between left and right children of a node, for the rest of the paper we will only consider automata that do not distinguish between left and right children, which means that we will be essentially working over unordered trees, since we will be ignoring the order of the children in the trees, and all of our results will be valid for unordered trees.

▶ **Definition 17.** *An NBUA of transition relation Δ is order-insensitive if for all transitions $(p, q, a, r) \in \Delta$, then $(q, p, a, r) \in \Delta$.*

▶ Remark 18. For a DBUA of transition map δ, it is order-insensitive if and only if $\delta_a(p, q) = \delta_a(q, p)$ for all states p, q and letter a.

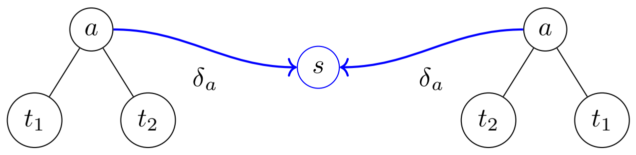


Figure 3 Illustration of order-insensitivity in a DBUA (see Remark 18). Both trees yield the same state s under the transition map δ_a.

▶ **Lemma 19.** *If $N = (\Sigma, Q, I, \Delta, F)$ an NBUA is order-insensitive, then* $\mathrm{Det}\,(N)$ *is also order-insensitive.*

Proof.

$$\begin{aligned}\delta_a(X,Y) &= \{q \in Q \mid (x,y,a,q) \in \Delta, x \in X, y \in Y\}\\ &= \{q \in Q \mid (y,x,a,q) \in \Delta, x \in X, y \in Y\} \quad (N \textit{ is order-insensitive})\\ &= \delta_a(Y,X)\end{aligned}$$

◀

Using both Theorem 16 and Lemma 19, we can conclude that order-insensitive NBUA and order-insensitive DBUA are equivalent. We will use them indiscriminately in the following sections, using the definition better suited to the context.

2.4 Universality

In this section, we define the reachability set of a DBUA and show that it can be used to decide universality, which is the property of accepting every tree.

▶ **Definition 20** (*reachability set*). *Let $A = (\Sigma, Q, \iota, \delta, F)$ a DBUA, and let $\mathrm{Reach}_A \subseteq Q$ be the smallest set such that:*
1. $\iota(a) \in \mathrm{Reach}_A$ *for all* $a \in \Sigma$*,*
2. $\delta(p,q,a) \in \mathrm{Reach}_A$ *for all* $p, q \in \mathrm{Reach}_A$ *and* $a \in \Sigma$*.*

This set captures all states that can be reached by the automaton starting from the leaves and combining them using the transition map. In particular, it represents the set of all states that can be reached when the automaton processes all possible trees, motivating the following lemma.

▶ **Lemma 21.** *Let A be a DBUA, then:*

$$\mathrm{Reach}_A = \{[\![A]\!](t) \mid t \in \mathrm{Tree}_\Sigma\}\,.$$

Proof.

We prove the equality by showing both inclusions.

- For the direction $\mathrm{Reach}_A \subseteq \{[\![A]\!](t) \mid t \in \mathrm{Tree}_\Sigma\}$, we proceed by induction over the construction of elements in Reach_A:
  - If $r = \iota(a)$ for some $a \in \Sigma$, then the tree a satisfies $[\![A]\!](a) = \iota(a) = r$.
  - If $r = \delta(p,q,a)$ where $p, q \in \mathrm{Reach}_A$, then by the induction hypothesis there exist $t', t'' \in \mathrm{Tree}_\Sigma$ such that $[\![A]\!](t') = p$ and $[\![A]\!](t'') = q$. Then the tree $t = a(t', t'')$ satisfies $[\![A]\!](t) = r$.
- For the direction $\{[\![A]\!](t) \mid t \in \mathrm{Tree}_\Sigma\} \subseteq \mathrm{Reach}_A$, we use structural induction on a tree t:
  - If $t = a$, then $[\![A]\!](a) = \iota(a) \in \mathrm{Reach}_A$.
  - If $t = a(t', t'')$, then by the induction hypothesis $[\![A]\!](t'), [\![A]\!](t'') \in \mathrm{Reach}_A$, so $[\![A]\!](t) = \delta([\![A]\!](t'), [\![A]\!](t''), a) \in \mathrm{Reach}_A$.

◀

▶ **Definition 22.** *Let A be a NBUA. We say that A is universal if it accepts every tree.*

We can use the reachability set to decide whether an automaton is universal. Intuitively, a DBUA is universal if the only states it can reach are final states. This is formalized in the following theorem.

▶ **Theorem 23.** *A DBUA $A = (\Sigma, Q, \iota, \delta, F)$ is universal if and only if $\mathrm{Reach}_A \subseteq F$.*

Proof.

- Suppose A is universal. Then for all $t \in \mathrm{Tree}_\Sigma$, we have $[\![A]\!](t) \in F$ because A accepts t. Since $\mathrm{Reach}_A = \{[\![A]\!](t) \mid t \in \mathrm{Tree}_\Sigma\}$, we conclude that $\mathrm{Reach}_A \subseteq F$.
- Conversely, suppose $\mathrm{Reach}_A \subseteq F$. For any tree $t \in \mathrm{Tree}_\Sigma$, we have $[\![A]\!](t) \in \mathrm{Reach}_A \subseteq F$. Hence, A accepts t and is therefore universal.

◀

▶ **Corollary 24.** *A saturation algorithm based on* Reach_A *decides universality.*

Proof. We can compute Reach_A using a saturation algorithm that starts with the initial states and iteratively adds states reachable by the transition map until no new states can be added. Once we have computed Reach_A, we can check if it is a subset of the final states. This algorithm will terminate because the number of states is finite, and it will decide universality by Theorem 23. ◀

This algorithm will be used later in Section 3 to decide uniformisation.

2.5 Monadic second-order logic on trees

In this subsection, we introduce monadic second-order logic (MSO) over finite binary trees, and examine its relationship with non-deterministic bottom-up automata, in particular with order-insensitive automata.

Our objective is to show that MSO and order-insensitive automata have the same expressive power, Theorem 31. This will allow us to reason about properties of MSO using automata theory techniques, which will be extensively used in Section 3.

▶ **Definition 25.** *Let $t \in \mathrm{Tree}_\Sigma$. We define a* model *of t as:*

$$\mathcal{M}(t) := (\mathrm{Nodes}(t), \sqsubseteq, (a(x))_{a\in\Sigma})$$

where each unary predicate $a(x)$ holds if and only if $t(x) = a$.

We write $t \models \varphi$ to mean that the formula φ holds in the model $\mathcal{M}(t)$.

▶ **Definition 26.** Monadic second-order logic *(*MSO*) is the extension of the first-order logic with the following features:*

- *Variables can be either first-order (denoting nodes) or second-order (denoting sets of nodes).*
- *Quantification can be done over both first-order and second-order variables.*
- *The logic includes the usual logical connectives and quantifiers.*
- *The* atomic formulas *include:*
  - *Equality between nodes.*
  - *$x \in X$ means that the node x belongs to the set X.*
  - *Ancestor relation: $x \sqsubseteq y$ means that x is an ancestor of y.*
  - *Unary predicates for each symbol in the alphabet: $a(x)$.*

▶ **Definition 27.** *We say that a language is* MSO-definable *if there exists an MSO-formula φ such that*

$$L = \{t \in \mathrm{Tree}_\Sigma \mid t \models \varphi\}.$$

We will discuss some examples of MSO formulas.

▶ **Example 28.** The formula $\text{leaf}(x) = \forall y,\ (\neg(x \sqsubseteq y) \vee x = y)$ expresses the fact that x is a leaf.

If x is a leaf, then it is not an ancestor of any other node except itself, which is captured by the formula.

▶ **Example 29.** The formula $\text{root}(x) = \forall y,\ x \sqsubseteq y$ defines the root of the tree, meaning a node that is an ancestor of every other node.

▶ **Example 30.** The following formula expresses that x and y are the immediate children of a node z:

$$\text{children}(x,y,z) = x \neq y \wedge z \sqsubseteq x \wedge z \sqsubseteq y \wedge \forall w\, ((z \sqsubseteq w \wedge (w \sqsubseteq x \vee w \sqsubseteq y)) \to (w = x \vee w = y))$$

We will now state the main result of this section, which establishes the equivalence between order-insensitive NBUA and MSO formulas.

▶ **Theorem 31.** *A language $L \subseteq \text{Tree}_\Sigma$ is MSO-definable if and only if there exists an order-insensitive NBUA A such that $\mathcal{L}(A) = L$.*

The proof follows the classical schema [1, 11, 7], but restricting to order-insensitive NBUA.

We start by showing that every automaton can be expressed as a formula. We will encode the existence of an accepting run of the automaton using second-order variables that assign states to nodes.

▶ **Lemma 32** (from automata to MSO). *If A is an order-insensitive NBUA, then there exists a MSO-formula φ such that for all $t \in \text{Tree}_\Sigma$, $t \models \varphi$ if and only if $t \in \mathcal{L}(A)$.*

Proof. Let $A = (\Sigma, Q, I, \Delta, F)$ be an order-insensitive NBUA, and let $k = |Q|$ be the number of states. We define the following MSO-formula φ.

$$\begin{aligned}
\varphi = \exists X_0 \dots \exists X_{k-1}\ \forall x & \left(\bigvee_{i=0}^{k-1} X_i(x) \right) && (1)\\
\wedge\ \forall x & \left(\bigwedge_{0 \le i < j \le k-1} \neg(X_i(x) \wedge X_j(x)) \right) && (2)\\
\wedge\ \forall x & \left(\text{leaf}(x) \to \bigvee_{(a,i) \in \mathcal{I}} X_i(x) \right) && (3)\\
\wedge\ \forall x \forall y \forall z & \left(\text{children}(x,y,z) \to \bigvee_{(p,q,a,r) \in \Delta} (X_p(x) \wedge X_q(y) \wedge a(z) \wedge X_r(z)) \right) && (4)\\
\wedge\ \forall x & \left(\text{root}(x) \to \bigvee_{q \in F} X_q(x) \right) && (5)
\end{aligned}$$

The sets X_i encode the run of the automaton: $x \in X_i$ means that the state i is assigned to the node x in some valid accepting run.

The formula ensures the following properties:

1 Every node is assigned some state.
2 No node is in more than one state.
3 Leaf nodes are assigned an initial state.

4 For each triple (x, y, z) forming a parent and its two children, there exists a transition in Δ from the children to the parent.

5 The root is labeled with a final state.

Since all of this proportier encode the existence of an accepting run of the automaton A over the tree t, we have that $t \models \varphi$ if and only if $t \in \mathcal{L}(A)$. ◀

We now turn to the other direction. Our goal is to show that for every MSO-formula over finite binary trees, there exists an equivalent order-insensitive non-deterministic bottom-up automaton.

The construction proceeds by induction on the structure of formulas, using standard constructions for closure and the atomic formulas. Throughout, we ensure that the resulting automata remain order-insensitive.

We begin with basic closure properties of such automata. We only give the definitions of the automata as the proofs are standard.

▶ **Lemma 33** (intersection). *Let $A_1 = (\Sigma, Q_1, I_1, \Delta_1, F_1)$ and $A_2 = (\Sigma, Q_2, I_2, \Delta_2, F_2)$ be two order-insensitive NBUAs. Then the product automaton $A_3 = (\Sigma, Q_1 \times Q_2, I_1 \times I_2, \Delta, F_1 \times F_2)$, where*

$$\Delta = \{((q_1, q_2), (p_1, p_2), (a_1, a_2), (r_1, r_2)) \mid (q_1, p_1, a_1, r_1) \in \Delta_1,\ (q_2, p_2, a_2, r_2) \in \Delta_2\},$$

is order-insensitive, and satisfies $\mathcal{L}(A_3) = \mathcal{L}(A_1) \cap \mathcal{L}(A_2)$.

▶ **Lemma 34** (union). *Let $A_1 = (\Sigma, Q_1, I_1, \Delta_1, F_1)$ and $A_2 = (\Sigma, Q_2, I_2, \Delta_2, F_2)$ be two order-insensitive NBUAs. Then the disjoint union $A_3 = (\Sigma, Q_1 \uplus Q_2, I_1 \uplus I_2, \Delta_1 \uplus \Delta_2, F_1 \uplus F_2)$, is also order-insensitive, and it accepts precisely the union of the two languages: $\mathcal{L}(A_3) = \mathcal{L}(A_1) \cup \mathcal{L}(A_2)$.*

▶ **Lemma 35** (complement). *Given an order-insensitive NBUA $A = (\Sigma, Q, \mathcal{I}, \Delta, F)$, the automaton*

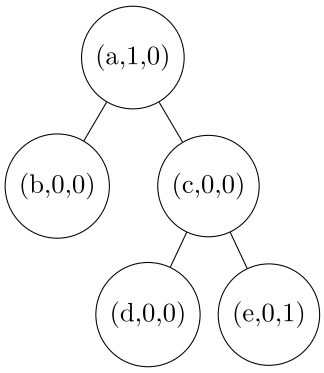
$$A' = (\Sigma, Q, \mathcal{I}, \Delta, Q \setminus F)$$

is also order-insensitive, and recognizes the complement of A: $\mathcal{L}(A') = \mathcal{L}(A)^{\complement}$.

We now turn to atomic formulas. These are handled by explicitly designed automata over expanded alphabets that encode variable assignments. We will only treat the ones that are relevant for our purposes, namely the atomic predicates $a(x)$ and $x \sqsubseteq y$.

In order to model variables we use annotated trees, where each node is annotated with a bit vector representing the assignment of variables. In this case, we will use the alphabet $\Sigma \times \{0, 1\}^n$, where n is the number of free variables in the formula.

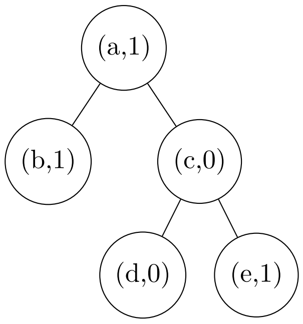
▶ **Example 36.** The following tree corresponds to the tree $a(b, c(d, e))$ with two free variables, where the first variable is assigned to the root and the second to the leaf with label e:



▶ **Definition 37** (annotated tree). *Given a set of nodes X and a tree t, we define an annotated tree as the pair (t, X). This pair corresponds to a new tree, where each label of a node is a pair consisting of the original label and a bit indicating whether the node belongs to the set X.*

We will use the notation $X(n)$, where n is a node of the tree, to denote the bit indicating whether n belongs to the set X.

▶ **Example 38.** Let t be the tree $a(b, c(d, e))$, and let $X = \{\varepsilon, 0, 11\}$. Then (t, X) corresponds to the annotated tree :



In the following, the transition map of the next automaton will be defined using a `match` syntax, where `_` denotes a wildcard that matches any value. Matches are evaluated from top to bottom, meaning the first rule that applies will be used.

▶ **Lemma 39.** *Let $a \in \Sigma$. Over the alphabet $\Sigma \times \{0, 1\}$, the order-insensitive automaton*

$$A = \big(\Sigma \times \{0,1\}, \{\text{Yes}, \text{No}\}, \iota, \delta, \{\text{Yes}\}\big)$$

defined as follows:

Initial states:

$$\begin{aligned} &\iota(a,1) && \Rightarrow \text{Yes} \\ &\iota(\_) && \Rightarrow \text{No} \end{aligned}$$

Transition map:

$$\begin{aligned} &\delta(\_,\_,(a,1)) && \Rightarrow \text{Yes} \\ &\delta(\text{No},\text{No},\_) && \Rightarrow \text{No} \\ &\delta(\_) && \Rightarrow \text{Yes} \end{aligned}$$

decides the atomic predicate $a(x)$, where x is a free first-order variable.

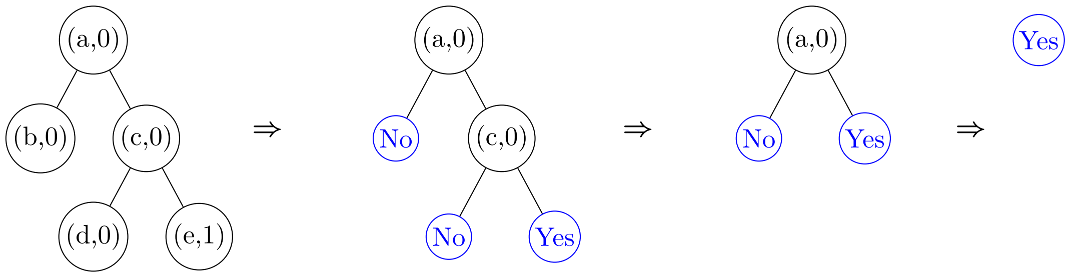


Figure 4 Execution of the automaton from Lemma 39 on the annotated tree $(a(b, c(d, e)), \{11\})$ for the label e.

▶ **Example 40.** Let t be the tree $a(b, c(d, e))$, and let A be the automaton from Lemma 39 for the letter e. Then the automaton A will accept the annotated tree $(t, \{11\})$ as the node 11 is labeled with e in t. The execution of the automaton is illustrated in Figure 4.

▶ **Lemma 41.** *Over the alphabet $\Sigma \times \{0,1\}^2$, the order-insensitive automaton*

$$A = \big(\Sigma \times \{0,1\}^2, \{\text{Yes}, \text{No}, \text{FoundY}\}, \iota, \delta, \{\text{Yes}\}\big)$$

defined as follows:

Initial states:

$\iota(\_,\_,1)$	$\Rightarrow$ FoundY
$\iota(\_)$	$\Rightarrow$ No

Transition map:

$\delta(\text{Yes},\_,\_)$	$\Rightarrow$ Yes
$\delta(\_,\text{Yes},\_)$	$\Rightarrow$ Yes
$\delta(\text{FoundY},\_,(\_,1,0))$	$\Rightarrow$ Yes
$\delta(\_,\text{FoundY},(\_,1,0))$	$\Rightarrow$ Yes
$\delta(\text{FoundY},\_,(\_,0,0))$	$\Rightarrow$ FoundY
$\delta(\_,\text{FoundY},(\_,0,0))$	$\Rightarrow$ FoundY
$\delta(\_,\_,(\_,0,1))$	$\Rightarrow$ FoundY
$\delta(\_)$	$\Rightarrow$ No

decides the atomic predicate $x \sqsubseteq y$, where x and y are free first-order variables.

▶ **Example 42.** Let t be the tree $a(b, c(d, e))$, and let A be the automaton from Lemma 41 for the atomic predicate $x \sqsubseteq y$, where x is the node labeled c at position 1 and y is the node labeled d at position 10. The execution of the automaton is illustrated in Figure 5.

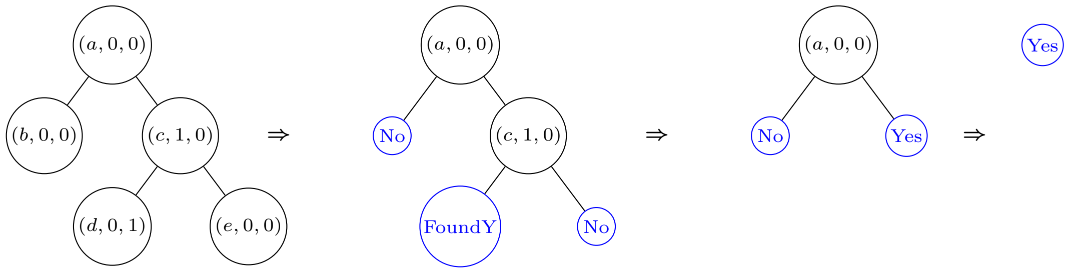


Figure 5 Execution of the automaton from Lemma 41 on the annotated tree $(a(b, c(d, e)), \{1\}, \{10\})$ for the nodes corresponding to the label c and e.

We now have all of the ingredients to construct an order-insensitive NBUA for any MSO-formula.

▶ **Lemma 43** (from MSO to automata). *Let φ be an MSO-formula. Then there exists an order-insensitive NBUA A such that for every $t \in \mathrm{Tree}_\Sigma$, we have:*

$$t \models \varphi \quad \textit{if and only if} \quad t \in \mathcal{L}(A).$$

Proof. The construction proceeds by structural induction on φ.

Atomic formulas such as $a(x)$ and $x \sqsubseteq y$ are handled directly by the automata in Lemmas 39 and 41, which operate over annotated trees with the appropriate bit vectors.

For conjunction and disjunction, we apply the induction hypothesis to obtain automata for the subformulas, then use the constructions from Lemmas 33 and 34 to obtain an automaton for the whole expression.

If $\varphi = \neg\psi$, then by the induction hypothesis we obtain an automaton for ψ. We apply Lemma 35 to obtain an automaton for φ.

Assume $\varphi = \exists X.\psi(X)$, where ψ has n free variables and X is the n-th one.

By the induction hypothesis, we have an order-insensitive DBUA (since avery NBUA can be converted to a DBUA) $A = (\Sigma \times \{0,1\}^n, Q, \iota, \delta, F)$ that accepts the language of $\psi(X)$.

We now construct an automaton that operates over $\Sigma \times \{0,1\}^{n-1}$, where X is no longer explicitly assigned. Instead, we simulate all possible annotations for X, and accept if at least one such annotation makes $\psi(X)$ true.

Define the new automaton $A' = (\Sigma \times \{0,1\}^{n-1}, \mathcal{P}(Q), \iota', \delta', F')$ where:

- Initial states:

  $$\iota'(a, \vec{b}) = \left\{\iota(a, \vec{b}, b') \mid b' \in \{0,1\}\right\}$$

  That is, we simulate both possibilities for the missing bit.
- Transition map:

  $$\delta'(P, Q, (a, \vec{b})) = \left\{\delta(p, q, (a, \vec{b}, b')) \mid p \in P,\ q \in Q,\ b' \in \{0,1\}\right\}$$

  Again, for every node guess if this it belongs to X, we compute the next state as if that guess were part of the input.
- Accepting states:

  $$F' = \{P \subseteq Q \mid P \cap F \neq \emptyset\}$$

  The automaton accepts if one of the runs under some assignment for X ends in an accepting state.

This automaton simulates all possible choices of a node for X, *i.e.* all possible bit annotations for the X-variable. Since A is order-insensitive, and the construction treats all nodes symmetrically, A' is also order-insensitive.

For $\varphi = \exists x.\psi(x)$, we have that it's a particular case of the previous construction.

Each inductive case preserves order-insensitivity, and constructs an NBUA for the given formula. Thus, by induction on the structure of φ, we obtain the desired automaton. ◀

3 Uniformisation

In this section we introduce the main concept of this work, uniformisation and we state and prove the main theorem, Theorem 46. Section 3.1 introduces the definitions and the main theorem, Section 3.2 defines the uniformiser automaton, and Section 3.3 and Section 3.4 prove the correctness and soundness of the algorithm, respectively.

3.1 Definitions and statement of the main theorem

We begin by formalising the core concepts.

▶ **Definition 44.** *An MSO-formula $\psi(X)$ is a uniformiser of an MSO-formula $\varphi(X)$ if, for all trees t, the following hold:*

1. *There exists at most one set X such that $t \models \psi(X)$.*
2. *If there exists a set X such that $t \models \varphi(X)$, then there exists a set Y such that $t \models \psi(Y) \wedge \varphi(Y)$.*

An MSO-formula $\varphi(X)$ is said to be uniformisable if it admits a uniformiser.

▶ Remark 45. An equivalent way to express that $\varphi(X)$ is uniformisable is that there exists an MSO-formula $\psi(x)$ such that, for all trees t:

$$t \models \exists X.\, \varphi(X) \quad \text{implies} \quad t \models \varphi(\{x \mid t \models \psi(x)\}).$$

Note that here we use a set defined by comprehension, which is not in the syntax of plain MSO logic. This is for commodity and can be easily translated into a syntactically correct equivalent statement.

We can now state the main result of this work.

▶ **Theorem 46.** *There exists an algorithm which, given an MSO-formula as input, decides whether it has a uniformiser over finite unordered binary trees.*

In fact, it is a consequence of a more precise result, Lemma 47, that we state now, though all the definitions necessary have not yet been provided.

▶ **Lemma 47.** *For all MSO-formulas $\varphi(X)$, the following statements are equivalent:*

1. *$\varphi(X)$ is uniformisable.*
2. *For all trees t that models $\varphi(X)$ for some X, t models $\varphi(X)$ for some definable X.*
3. *For all trees t that models $\varphi(X)$ for some X, t models $\varphi(X)$ for some X invariant under the automorphisms of t (i.e. such that $\sigma(X) = X$ for all $\sigma \in \mathrm{Aut}(t)$).*
4. *The uniformiser automaton of $\varphi(X)$ is universal.*

As as consequence, uniformisability of MSO over trees is decidable.

In the rest of this section, we recall the definitions of definable sets and invariance under automorphisms from which we immediately get the implications from Item 1 to Item 2, and from Item 2 to Item 3. The uniformiser automaton is then defined in the next section, Definition 52. The implication from Item 3 to Item 4 is then the subject of Lemma 56 in Section 3.4 and the implication from Item 4 to Item 1 is stated in Lemma 54, Section 3.3.

▶ **Definition 48** (definable set)**.** *Let t be a tree. A set $X \subseteq \mathrm{Nodes}\,(t)$ is definable if there exists an MSO-formula $\varphi(x)$ such that $X = \{x \mid t \models \varphi(x)\}$.*

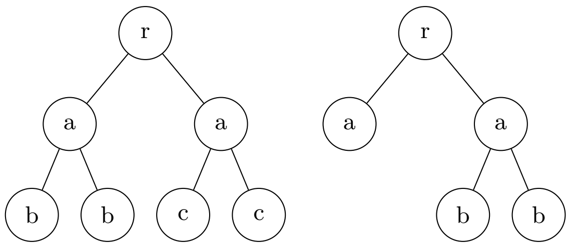
In particular, if $\varphi(X)$ is uniformisable, there exists a uniformiser $\psi(x)$ such that for all trees t, we have that the set $X_d = \{x \mid t \models \psi(x)\}$ is definable in t and models $\varphi(X_d)$ if t models $\exists X.\, \varphi(X)$, see Remark 45. This proves the implication from Item 1 to Item 2 in Lemma 47.

Another concept used in this section is the one of automorphisms of trees.

▶ **Definition 49** (automorphism)**.** *An automorphism of a tree t is a bijection $\sigma : \mathrm{Nodes}\,(t) \to \mathrm{Nodes}\,(t)$ preserving the tree structure: if u is a child of v, then $\sigma(u)$ is a child of $\sigma(v)$, and σ preserves the labels of nodes. The group of all automorphisms of t is denoted $\mathrm{Aut}(t)$.*

In other words, an automorphism is a symmetry of the tree that rearranges its nodes while preserving the ancestor relation and labels. Let us illustrate this concept with an example.

▶ **Example 50.** We compare the automorphism groups of two binary trees.



Tree A: Symmetric **Tree B: Asymmetric**

Tree A is a complete binary tree of height 2. In this case, there are 8 automorphisms, corresponding to all possible ways of swapping left and right children at each level.

Tree B, on the other hand, is asymmetric. The left child of the root is a leaf, while the right child is an internal node with two children. Since these subtrees are not structurally the same, the only automorphisms are:

- the identity (doing nothing), and
- swapping the two children of the right subtree.

The next lemma states an intuitive property of definable sets in relation to automorphisms, we propose an intuitive proof idea.

▶ **Lemma 51.** *If X is definable in a tree t, then $\sigma(X) = X$ for all $\sigma \in \mathrm{Aut}(t)$.*

Idea of a proof. If X is definable, there exists an MSO-formula $\varphi(x)$ such that $X = \{x \mid t \models \varphi(x)\}$. We want to show that for all $\sigma \in \mathrm{Aut}(t)$ and all $x \in X$, we have $t \models \varphi(\sigma(x))$. Since σ is a bijection, this suffices to conclude $\sigma(X) = X$.

The idea is that an MSO-formula, by construction, cannot distinguish between nodes that are mapped to each other by automorphisms, since the only relationship between nodes that an MSO-formula can express is the ancestor relation, which is preserved by automorphisms.

The proof proceeds by induction on the structure of φ. Atomic formulas are invariant under automorphisms by definition, since automorphisms preserve the tree structure and labeling. The logical connectives (conjunction, disjunction, negation) and quantifiers preserve this invariance, so the property holds inductively.

Therefore, for all $x \in X$, we have $t \models \varphi(x)$ implies $t \models \varphi(\sigma(x))$, and hence $\sigma(x) \in X$. Thus, $\sigma(X) = X$. ◀

The implication from Item 2 to Item 3 in Lemma 47 is a direct consequence of Lemma 51.

3.2 The uniformiser automaton

Let us fix an MSO-formula $\varphi(X)$. At very high level, what we aim at is a procedure that would solve a statement of the form[1]:

$$(\exists X.\, \varphi(X)) \to (\exists X.\, \varphi(X) \text{ and } X \text{ is invariant under automorphism}).$$

[1] This is an intuition, and not a valid formula of MSO.

We aim to build an automaton that somehow accepts the trees for which this property holds. This is the uniformiser automaton for the formula, that we describe below. We then show in Lemma 54 that if it is universal, then $\varphi(X)$ is uniformisable, and in Lemma 56 that if it is not universal, then there exists a tree t that models $\varphi(X)$ for some X, but does not model $\varphi(X)$ for some X that would be invariant under the automorphisms of t.

▶ **Definition 52** (uniformiser automaton). *Let $\varphi(X)$ be an MSO-formula and $A = (\Sigma \times \{0,1\}, Q, \iota, \delta, F)$ its DBUA. The uniformiser automaton is defined as*

$$U = (\Sigma, \mathcal{P}(Q) \times \mathcal{P}(Q), \iota_U, \delta_U, F')$$

where:

- $\iota_U(a) = (\iota(a,*), \iota(a,*))$
- $\delta_U((X,X'),(Y,Y'),a) = (\delta(X,Y,(a,*)), \delta_{U_2}(X',Y',a))$ *with:*

$$\delta_{U_2}(X',Y',a) = \begin{cases} \{\delta(p,p,(a,b)) \mid p \in X', b \in \{0,1\}\} & \textit{if } X = Y \textit{ and } X' = Y' \\ \delta(X',Y',(a,*)) & \textit{otherwise} \end{cases}$$

- $F' = \{(X,X') \mid (X \cap F \neq \emptyset) \Rightarrow (X' \cap F \neq \emptyset)\}$

Using the notation $\delta(X,Y,(a,*))$ for $\bigcup_{b\in\{0,1\}} \delta(X,Y,(a,b))$, and $\iota(a,*)$ for $\bigcup_{b\in\{0,1\}} \iota(a,b)$. Note that in this definition, the automaton A corresponds to the automaton for the formula $\varphi(X)$, particular if A accepts a tree t then there exists a set X such that $t \models \varphi(X)$.

Intuitively, U checks whether $\varphi(X)$ has a witness, and attempts to reconstruct a witness invariant under automorphisms using symmetry constraints.

Each state of the uniformiser automaton is a pair of states of the original automaton for $\exists X.\varphi(X)$. The first component represents a standard run of said automaton (checking if the formula $\varphi(X)$ can be modeled), while the second component simulates a restricted run that captures a solution that is invariant under automorphisms. When both sides behave identically on a given subtree, that is, when the shape and labels of the tree do not allow us to distinguish between them, the automaton enforces that they must transition identically, ensuring symmetry. This restriction is what guarantees that the selected set X is invariant under automorphisms.

▶ **Definition 53** (uniformisability checking algorithm). *Given an MSO-formula $\varphi(X)$ as an input, the uniformisability checking algorithm constructs the uniformiser automaton for $\varphi(X)$ and checks whether it is universal, using the algorithm from Corollary 24. If it is universal, then the algorithm returns 'yes', indicating that $\varphi(X)$ is uniformisable. Otherwise, it returns 'no', indicating that $\varphi(X)$ is not uniformisable.*

In the rest of this section, we will prove that this algorithm is correct and sound, leading to the main result of this article, Theorem 46.

3.3 Correctness

In this section, we prove the correctness of the uniformisability checking algorithm, which is the implication from Item 4 to Item 1 in Lemma 47. This is what states the next lemma.

▶ **Lemma 54.** *Let $\varphi(X)$ be an MSO-formula, if the uniformiser automaton of $\varphi(X)$ is universal then $\varphi(X)$ is uniformisable.*

Proof. Let $A = (\Sigma \times \{0,1\}, Q, \iota, \delta, F)$ be the automaton for $\varphi(X)$ and lets suppose that $U = (\Sigma, \mathcal{P}(Q) \times \mathcal{P}(Q), \iota', \delta', F')$ the uniformiser automaton of $\varphi(X)$ is universal. We need to exhibit a uniformiser for $\varphi(X)$.

We start by introducing a set of choice functions, their purpose is that, given a set of states of A, they will choose a particular state of A following some restrictions. This means that we will have a way to reconstruct a run of A over a tree t annotated with a particular set X of nodes of t, that satisfies the restrictions of δ_{U_2}, and that is unique.

- We first introduce choiceNodes : $\mathcal{P}(Q) \times \mathcal{P}(Q) \times Q \times \Sigma \to Q \times Q \times \{0,1\}$ that given (X', Y', r, a) chooses a triple (b, p, q) such that, if $r \in \delta_{U_2}(X', Y', a)$ then $\delta(p, q, (a, b)) = r$ where $p \in X', q \in Y'$ and if $X = Y$ and $X' = Y'$ then $p = q$. This function will select the next state when we are at an internal node of the tree.
- We also introduce choiceLeaves : $Q \times \Sigma \to \{0,1\}$, which chooses b such that $p = \iota((a,b))$. This function will select the next state when we are at a leaf of the tree.
- Finally, we need to choose from a final state: choiceFinal : $\mathcal{P}(Q) \to Q$ where choiceFinal$(X) = q$ such that if $X \cap F \neq \emptyset$ then $q \in X \cap F$. This will correspond to the final state of our run.

If the premises are not satisfied in any of the cases, then these choice functions return an arbitrary value. Since th number of states is finite, such choice functions exist for an arbitrary ordering of the states of A.

Now, thanks to the choice function, we can define a function ρ_U : Nodes$(t) \to Q$ that corresponds to a run A over t annotated with a set X_U of nodes defined later. The fact that it is a run will be proved later.

We define the set X_U and the ρ_U by induction on the depth of the node as follows:

- For the root, we define $\rho_U(\varepsilon) = \text{choiceFinal}(\delta_{U_2}(t))$.
- For all internal nodes x with children y and z.
  Let $(p, q, v) = \text{choiceNodes}(\delta_{U_2}(t^y), \delta_{U_2}(t^z), \rho_U(x), t(x))$, we define $\rho_U(y) = p$, $\rho_U(z) = q$, and $x \in X_U$ if and only if $b = 1$.
- For all leaves x, let b be $b = \text{choiceLeaves}(p, t(x))$ we define $\rho_U(x) = p$, where $x \in X_U$ if and only if $b = 1$.

▷ Claim 55. We claim now that the ρ_U is well defined, unique, and indeed a run of A over (t, X_U).

Proof of the claim. The proof proceeds by top-down induction.

- For ε, the root of t, we have that $\rho_U(\varepsilon) = \text{choiceFinal}(\delta_{U_2}(t))$, since choiceFinal is deterministic, $\rho_U(\varepsilon)$ is unique.
- For the internal nodes x with children y and z, there exists $r \in Q$, such that $\rho_U(x) = r$ by the induction hypothesis. Then, since choiceNodes is deterministic, there exists a unique tuple (p, q, b) such that $(p, q, b) = \text{choiceNodes}(\delta_{U_2}(t^y), \delta_{U_2}(t^z), r, t(x))$. We have that $\rho_U(y) = p$ and $\rho_U(z) = q$, which verify that $\delta(p, q, (t(x), b)) = r$. We also have that $x \in X_U$ if and only if $b = 1$. This is then a valid run of A over (t, X_U)
- For a leaf x of label a, by the induction hypothesis we have that $b = \text{choiceLeaves}(a, \rho_U(x))$ which is well defined since $a \in \Sigma$ and unique. We also have that $x \in X_U$ if and only if $b = 1$ and $\rho(x) = \iota((a, b))$.

The claim is proved. ◀

In particular, it is a run that satisfies the restrictions of δ_{U_2} and so if $\rho_U(x) \in F$ then X_U models $\varphi(X)$.

We are now ready to define an MSO-formula unif(X) that witnesses that $\varphi(X)$ is uniformisable. This formula simply states using the MSO-syntax the existence of the run ρ_U described above, *i.e.* $t \models \text{unif}(X)$ if and only if $X = X_U$.

Given a set of states P, from Theorem 31 there exists a formula u_state$_P(x)$ such that $t \models \text{u\_state}_P(x)$ if and only if $\delta_{U_2}(t^x) = P$.

Let $k = |Q|$, we will define 5 subformulas of unif(X).

$$\begin{aligned}\text{single\_state}(X_{p_1}, \ldots, X_{p_k}) &= \forall x, \bigvee_{1 \le i \le k} X_{p_i}(x) \\ &\wedge \forall x, \bigwedge_{1 \le i,j \le k, i \neq j} \neg(X_{p_i}(x) \wedge X_{p_j}(x))\end{aligned}$$

This MSO-formula states that every node is assigned to exactly one of the X_{p_i}, which are the states of the automaton A.

$$\text{root\_node}(X_{p_1}, \ldots, X_{p_k}) = \forall x, \text{root}(x) \to \Big(\bigvee_{\left(\substack{p_i \in \\ \delta_{U_2}(\varepsilon)}\right)} X_{p_i}(x)\Big)$$

We now state that the root of the tree is assigned to the state chosen by choiceFinal.

$$\text{in\_set}_b(X, x) = (b = 1) \Leftrightarrow X(x)$$

This formula can be expressed in real MSO syntax, but we propose a more intuitive definition.

$$\text{leaf\_node}(X_{p_1}, \ldots, X_{p_k}) = \forall x \left(\text{leaf}(x) \to \bigvee_{\substack{(a,q) \in \iota, \\ \text{choiceLeaves}(a,q) = b}} X_i(x) \wedge \text{in\_set}_b(X, x) \right)$$

This MSO-formula verifies that the state of a leaf is the one chosen by choiceLeaves, and that the leaf is in the set X if and only if $b = 1$.

$$\begin{aligned}\text{internal\_nodes}(X_{p_1}, \ldots, X_{p_k}, X) &= \forall x \forall y \forall z, \text{children}(x, y, z) \to \\ &\bigvee_{\substack{(P',Q',a,R') \in \delta_{U_2}, r \in R', \\ \text{choiceNodes}(P',Q',a,r) = (p,q,b)}} \Big(\text{u\_state}_{P'}(x) \wedge \text{u\_state}_{Q'}(y) \\ &\wedge X_p(x) \wedge X_q(y) \wedge a(z) \wedge X_r(z) \wedge \text{in\_set}_b(X, x)\Big)\end{aligned}$$

That last formula states that for every internal node x with children y and z, the state of x is the one chosen by choiceNodes, that the states of y and z are the ones chosen by choiceNodes, and that node is in the set X if and only if $b = 1$.

We can now define unif(X):

$$\begin{aligned}\text{unif}(X) &= \exists X_{p_1}, \ldots, \exists X_{p_k}, \text{single\_state}(X_{p_1}, \ldots, X_{p_k}) \\ &\wedge \text{leaf\_node}(X_{p_1}, \ldots, X_{p_k}) \\ &\wedge \text{root\_node}(X_{p_1}, \ldots, X_{p_k}) \\ &\wedge \text{internal\_nodes}(X_{p_1}, \ldots, X_{p_k}, X)\end{aligned}$$

$\mathrm{unif}(X)$ describes the run ρ_U over (t, X_U) and has a unique solution, X_U.

Lets show that $\mathrm{unif}(X)$ is a uniformiser for $\varphi(X)$. Since U is universal the we have that, whenever $\varphi(X)$ has a solution for a tree t, then there exists an accepting run ρ of U over t such that $\rho(\varepsilon) = (X, X')$ and such that $X \cap F \neq \emptyset$. Since the run is accepting $X' \cap F \neq \emptyset$, and we have that ρ_U is then well defined and $\mathrm{unif}(X)$ has a solution. Since ρ_U is a run of A, then we have that X_U is also a solution of $\varphi(X)$, an since $\mathrm{unif}(X)$ as a unique solution, we have that $\mathrm{unif}(X)$ is indeed a uniformiser for $\varphi(X)$.

◀

3.4 Soundness

In this section, we prove the soundness of the uniformisability checking algorithm, by compleating the proof of Item 3 to Item 4 in Lemma 47 by contraposition. This chain of implications states that if $\varphi(X)$ is uniformisable, then the uniformiser automaton of $\varphi(X)$ is universal, *i.e.* that the algorithm is sound.

▶ **Lemma 56.** *Let $\varphi(X)$ be an MSO-formula, if the uniformiser automaton of $\varphi(X)$ is not universal, then there exists a tree t that models $\varphi(X)$ for some X, but does not model $\varphi(X)$ for some X that would be invariant under the automorphisms of t.*

Proof. Let $A = (\Sigma \times \{0,1\}, Q, \iota, \delta, F)$ be the automaton for $\varphi(X)$ and lets suppose that $U = (\Sigma, \mathcal{P}(Q) \times \mathcal{P}(Q), \iota', \delta', F')$ the uniformiser automaton of $\varphi(X)$ is not universal. We will show that there exists a tree t that models $\varphi(X)$ for some X, but does not model $\varphi(X)$ for some X that is invariant under the automorphisms of t.

Since U is not universal, there exists $(R, R') \in \mathrm{Reach}_U$, such that $R \cap F \neq \emptyset$ and $R' \cap F = \emptyset$. We want to define a tree $t_{R,R'}$ such that

1. $R \subseteq \{\delta_A(t_{R,R'}, X) \mid X \subseteq \mathrm{Nodes}(t_{R,R'})\}$
2. $\{\delta_A(t_{R,R'}, X) \mid \forall \sigma \in \mathrm{Aut}(t_{R,R'}), \sigma(X) = X\} \subseteq R'$
3. $[\![U]\!](t_{R,R'}) = (R, R')$

We proceed by induction over the construction of elements in Reach_U:

If $(T, T') = (\iota(a,*), \iota(a,*))$ for $a \in \Sigma$, then Item 1 and Item 3 are trivially satisfied and we have that $\mathrm{Aut}(a) = \{\mathrm{id}\}$, so Item 2 is also satisfied.

If $(T, T') = \delta_U((P, P'), (Q, Q'), a)$ for $a \in \Sigma$ and $P, P', Q, Q' \in \mathrm{Reach}_U$, then by the induction hypothesis there exists $t_{P,P'}$ and $t_{Q,Q'}$ that verify the hypothesis. We can construct $t_{T,T'} = a(t_{P,P'}, t_{Q,Q'})$. Lets prove that it satisfies the three properties.

We will prove Item 3. By definition of $[\![U]\!](\cdot)$, $[\![U]\!](t_{T,T'}) = \delta_U([\![U]\!](t_{P,P'}), [\![U]\!](t_{Q,Q'}), a)$. If we use Item 3 of the induction hypothesis we deduce that $\delta_U([\![U]\!](t_{P,P'}), [\![U]\!](t_{Q,Q'}), a) = \delta_U((P, P'), (Q, Q'), a)$ And by definition of (T, T'), $\delta_U((P, P'), (Q, Q'), a) = (T, T')$. Item 3 is then verified.

By definition, $T = \delta_A(P, Q, (a, *))$. Since by Item 1 of the induction hypothesis $P \subseteq \{\delta_A(t_{P,P'}, X) \mid X \subseteq \mathrm{Nodes}(t_{P,P'})\}$ and $Q \subseteq \{\delta_A(t_{Q,Q'}, X) \mid X \subseteq \mathrm{Nodes}(t_{Q,Q'})\}$ we have that $T = \delta_A(P, Q, (a, *)) \subseteq \{\delta_A(t_{T,T'}, X) \mid X \subseteq \mathrm{Nodes}(t_{T,T'})\}$. We have proven Item 1.

We will prove Item 2 by case distinction. Let $X \subseteq \mathrm{Nodes}(t_{P,P'})$ such that $\forall \sigma \in \mathrm{Aut}(t_{T,T'}), \sigma(X) = X$, we introduce the sets $X_1 = X \cap \mathrm{Nodes}(t_{P,P'})$ and $X_2 = X \cap \mathrm{Nodes}(t_{Q,Q'})$.

- Lets suppose $t_{P,P'} \cong t_{Q,Q'}$. By Item 3 of the induction hypothesis $[\![U]\!](t_{P,P'}) = (P, P')$ and $[\![U]\!](t_{Q,Q'}) = (Q, Q')$, since $[\![U]\!](\cdot)$ is a function and $[\![U]\!](t_{P,P'}) = [\![U]\!](t_{Q,Q'})$, we have that $P = Q$ and $P' = Q'$.

Let σ be the automorphism over Nodes (T, T') that swaps $t_{P,P'}$ and $t_{Q,Q'}$. We have that $\sigma(X) = X$. Since $\delta(X) = X$, δ must map X_1 to X_2, and the same is true for X_2. So we have that $X_1 = X_2$ which means that $\sigma(X_1) = X_1$ and $\sigma(X_2) = X_2$. By construction we have that $p := \delta_A(t_{P,P'}, X_1) = \delta_A(t_{Q,Q'}, X_2)$. Using Item 2 of the induction hypothesis, we find that $p \in P'$. So $\delta_A(t_{T,T'}, X) = \delta_A(p, p, (a, X(\text{root}))) \in T'$.

- Lets suppose $t_{P,P'} \ncong t_{Q,Q'}$, in this case we have that $P' \neq Q'$ or $P \neq Q$ and thus $T' = \delta_{U_2}(P', Q', a) = \delta_a(P', Q', (a, *))$.
  We have to prove that $\{\delta_A(t_{T,T'}, X) \mid \forall\sigma \in \text{Aut}(t_{T,T'}), \sigma(X) = X\} \subseteq T'$. Let $\sigma \in \text{Aut}(t_{T,T'})$. Since $t_{P,P'} \ncong t_{Q,Q'}$, σ cannot swap $t_{P,P'}$ and $t_{Q,Q'}$. Since we have that $\sigma(X) = X$, then we deduce that $\sigma(X_1) = X_1$ and $\sigma(X_2) = X_2$, because σ must act on the subtrees independently (because the subtrees are not isomorphic). Let $p := \delta_A(t_{P,P'}, X_1)$ and $q := \delta_A(t_{Q,Q'}, X_2)$. Using Item 2 of the induction hypothesis we deduce that $p \in P'$ and $q \in Q'$. So $\delta_A(t_{T,T'}, X) = \delta_A(p, q, (a, X(\text{root}))) \in T'$.

We have proved the three properties.

Lets prove the lemma by exhibiting that no X verifying $t_{R,R'} \models \varphi(X)$ is invariant under the automorphisms of $t_{R,R'}$. Since $R \cap F \neq \emptyset$, there exists X that models $\varphi(X)$ and belonging to $R \cap F$. We can use Item 1 to deduce that $X \in \{\delta_A(t_{R,R'}, X) \mid X \subseteq \text{Nodes}\,(t_{R,R'})\} \cap F$. We also know that $R' \cap F = \emptyset$, we can the deduce that $\{\delta_A(t_{R,R'}, X) \mid \forall\sigma \in \text{Aut}(t_{R,R'}), \sigma(X) = X\} \cap F = \emptyset$ thanks to Item 2. So we know that $X \notin \{\delta_A(t_{R,R'}, X) \mid \forall\sigma \in \text{Aut}(t_{R,R'}), \sigma(X) = X\} \cap F = \emptyset$ which means that no X satisfying $t_{P,P'} \models \varphi(X)$ is invariant under the automorphisms of $t_{R,R'}$. ◀

Lemma 47 is then proved. We can conclue with a simple proof of our main theorem.

Proof of Theorem 46. Lemma 47 states the equivalence between the uniformisability of a formula $\varphi(X)$ and the universality of its uniformiser automaton U. We can use the uniformisability checking algorithm to construct U and check whether it is universal. This algorithm decides whether $\varphi(X)$ is uniformisable. ◀

4 Conclusion

We proved that the uniformisation problem for MSO formulas over unordered finite binary trees is decidable, using automata theory based on order-insensitive bottom-up automata. The construction also gives a way to extract a uniformiser when one exists. Along the way, we showed that uniformisability coincides with definability and automorphism invariance in this setting.

There are several interesting directions for future work. While the decidability of the general case for infinite trees is a hard open question, it could be worth investigating whether this approach could help identify subclasses of infinite trees for which uniformisability is decidable. Similarly, it would be natural to consider extending these results to unranked trees, where nodes can have an arbitrary number of children.

References

1 J. Richard Büchi. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6(1-6):66–92, 1960. URL: `https://onlinelibrary.wiley.com/doi/abs/10.1002/malq.19600060105`, `arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/malq.19600060105`, `doi:10.1002/malq.19600060105`.

2 Arnaud Carayol and Christof Löding. *Uniformization in Automata Theory.* 01 2015.

3 Christian Choffrut and Serge Grigorieff. *Uniformization of Rational Relations*, pages 59–71. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. doi:10.1007/978-3-642-60207-8_6.

4 Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Christof Löding, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*. 2008. URL: https://inria.hal.science/hal-03367725.

5 John Doner. Tree acceptors and some of their applications. *Journal of Computer and System Sciences*, 4(5):406–451, 1970. URL: https://www.sciencedirect.com/science/article/pii/S0022000070800411, doi:10.1016/S0022-0000(70)80041-1.

6 Yuri Gurevich and Saharon Shelah. Rabin's uniformization problem. *Journal of Symbolic Logic*, 48, 12 1983. doi:10.2307/2273673.

7 Jean-Éric Pin, editor. *Handbook of Automata Theory*, chapter 7. European Mathematical Society Publishing House, Zürich, Switzerland, 2021. URL: https://doi.org/10.4171/Automata, doi:10.4171/AUTOMATA.

8 Alexander Moshe Rabinovich. On decidability of monadic logic of order over the naturals extended by monadic predicates. *Inf. Comput.*, 205:870–889, 2007. URL: https://api.semanticscholar.org/CorpusID:15159859.

9 Dirk Siefkes. J. w. thatcher and j. b. wright. generalized finite automata theory with an application to a decision problem of second-order logic. mathematical systems theory, vol. 2 (1968), pp. 57–81. *Journal of Symbolic Logic*, 37(3):619–620, 1968. doi:10.2307/2272790.

10 Dirk Siefkes. The recursive sets in certain monadic second order fragments of arithmetic. *Archiv für mathematische Logik und Grundlagenforschung*, 17(1):71–80, Mar 1975. doi:10.1007/BF02280817.

11 Wolfgang Thomas. *Languages, Automata, and Logic*, pages 389–455. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. doi:10.1007/978-3-642-59126-6_7.